PREMIERS PAS EN PROGRAMMATION

SUR ORIC

Georges Viguier

Edimicro

PREMIERS PAS EN PROGRAMMATION

SUR ORIC

© F.D.S./Edimicro 1983
Première édition

Imprimé en France. Droits mondiaux réservés.

« La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toutes représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article 40). »
« Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal. »

ISBN : 2-904457-07-0

EDIMICRO

DÉPARTEMENT ÉDITIONS DE F.D.S. SARL

121/127, avenue d'Italie, 75013 Paris.

PREMIERS PAS EN PROGRAMMATION SUR ORIC

par

Georges VIGUIER

Edimicro

Préface

Ceux qui ne croient pas en l'ordinateur en 1984 sont ceux qui ne croyaient pas en l'automobile en 1884. Utilisons l'ordinateur comme nous utilisons notre voiture : qu'importe la façon dont ces objets de consommation sont fabriqués. L'important, est de savoir les piloter afin de bien s'en servir. Il y aura demain deux catégories d'individus : ceux qui sauront, et ceux qui ne sauront pas programmer. Le bon sens et les événements nous prouvent qu'il vaut mieux être dans la première catégorie.

Dans quelques années ce sera la clé d'accès au savoir, et à la formidable puissance qu'offriront de gigantesques centres informatique. Il faut, dès maintenant, se préparer à cette véritable révolution industrielle.

Le Professeur A. P. ERSHOV l'a bien écrit :

« L'ordinateur sera bien plus qu'un outil de calcul : il apporte un nouvel « environnement intellectuel, un cadre opérationnel que l'homme exploitera « naturellement dans sa vie quotidienne. Ce faisant, il réduira son temps de « formation, augmentera son activité et se préparera à participer par son « travail à la seconde révolution industrielle précipitée par l'ordinateur « et l'automation.

« En d'autres termes, la programmation est une seconde alphabétisation. »

Aujourd'hui, l'informatique est mon métier. C'est aussi mon hobby. Programmer est devenu pour moi une véritable passion dont j'aurais beaucoup de mal à me priver. Je dois vous avouer aussi mon goût prononcé pour l'enseignement : j'aime à faire savoir aux autres ce que j'ai appris. C'est une des principales raisons qui m'a poussé à écrire cet ouvrage où j'espère vous convaincre que la programmation n'a rien de redoutable, bien au contraire, et vous en donner le goût et le plaisir...

Georges VIGUIER

Table des matières

Avant-Propos

CHAPITRE 1. — **Généralités**

1.1 Du boulier au microprocesseur 3
1.2 Voies de développement 5
1.3 Domaines de l'informatique 5
1.3.1 Informatique scientifique 6
1.3.2 Informatique de gestion 7
1.3.3 Informatique industrielle 8
1.3.4 L'informatique médicale 9
1.3.5 L'éducation 10
1.3.6 Informatique de bureau 10
1.3.7 Informatique personnelle 11

CHAPITRE 2. — **Le calculateur numérique ou l'ordinateur**

2.1 Qu'est-ce qu'un ordinateur ? 15
2.2 Que trouve-t-on dans un ordinateur 15
2.3 Fonctions principales de l'ordinateur 16
2.4 Les mémoires 17
2.4.1 La technologie des mémoires 18
2.4.2 Que contient la mémoire ? 20
2.4.3 Le binaire : quelques révisions 24
2.5 L'unité centrale 28
2.6 Temps de traitement 30
2.7 Les organes d'entrée/sortie 31
2.8 Le software 33
2.8.1 Les outils d'aide au développement 34
2.8.2 Les langages 34
2.8.3 Les outils d'exploitation 35

CHAPITRE 3. — **Une approche de l'analyse**

CHAPITRE 4. — **Introduction à la programmation**

4.1 Introduction à la programmation 45
4.2 La programmation à la portée de tous 45
4.3 Le calculateur humain 46
4.4 Exercice n° 1 : Amusez-vous à être l'ordinateur 47
4.5 Comment faire un organigramme ? 48
4.5.1 Qu'est-ce qu'une expression simple ? 51
4.5.2 Qu'est-ce qu'une expression conditionnelle ? 52
4.5.2.1 TANT QUE (expression simple) FAIRE (expression simple) 53
4.5.2.2 SI (expression simple) ALORS (expression simple) 54
4.5.2.3 POUR (x) COMMENÇANT A (i) ET ALLANT JUSQU'A (n) ALORS 55
4.6 Exercice n° 2 55
4.6.1 Les aiguillages ou drapeaux 57
4.7 Exercice n° 3 : le tri 58

CHAPITRE 5. — **De la théorie à la pratique**

5.1 Avant-propos 65
5.2 Un bon conseil 65
5.3 Rappel 66
5.4 Reprenons l'exercice n° 1 67
5.5 Passons à l'exercice n° 2 72
5.6 Exercice n° 3 74

CHAPITRE 6. — **Du fonctionnel à la programmation**

6.1 Les impôts 81
6.2 Le loto 88
6.3 Préfixes et radicaux 93
6.4 Les nombres premiers 99
6.5 Jeu de cartes 103

CHAPITRE 7. — **La programmation structurée**

7.1 Avant-propos 127
7.2 Situation actuelle 128
7.3 Historique 130
7.4 Vingt règles pour bien programmer 134

Annexes

A. Conversion d'un nombre décimal vers une base « r » 139
B. Conversion d'un nombre d'un système de base « r » vers le décimal 140
C. Tableau de conversion décimal-binaire des 256 premiers entiers positifs 142
D. Tableau de conversion décimal-hexadécimal des 512 premiers entiers positifs 144
E. Codes ASCII 146
F. Rappel des principales instructions du langage BASIC 148
G. Différents symboles utilisables pour l'organigramme 152

Conclusion

CHAPITRE **1**

Généralités

1.1 DU BOULIER AU MICROPROCESSEUR

Les origines de l'informatique remontent à l'invention des nombres, à la représentation des informations sous forme numérique, et au développement de machines automatiques.

La première machine à calculer a probablement été le **BOULIER**. Développé en Chine vers l'an — 500 av. J.-C., il est toujours utilisé !

Il faut attendre ensuite 1617 pour que John Napier montre comment faire des multiplications. En 1642, Blaise Pascal met au point sa célèbre calculatrice numérique.

Au XVIIe siècle, Jacquard invente un métier à tisser programmé avec des cartes perforées.

Le mathématicien, physicien et philosophe **Leibnitz** imagine un système de numération à base 2. C'est l'inventeur du **calcul binaire.**

De 1815 à 1864, Georges Boole, mathématicien, invente une algèbre fondée sur une logique à base 2. Pour cela il reprend les travaux de Leibnitz. C'est l'un des promoteurs des mathématiques contemporaines.

En 1833, Charles Barrage marie les inventions de Jacquard et de Pascal et propose une machine analytique, trop complexe cependant pour l'époque.

Moins élaborée, la première machine électromécanique est réalisée par H. Hollerith. Elle travaille avec des cartes perforées et servira lors du recensement des Américains en 1885.

Tout au début de ce siècle, on verra ensuite apparaître des équipements industriels à cartes perforées — pour travaux comptables et statistiques — fabriqués par l'ingénieur américain Powers.

C'est à cette même époque que Thomas Watson crée IBM en reprenant la Société TABULATING MACHINES CO.

En 1924, l'Ingénieur norvégien Bull dépose des brevets sur un matériel électromécanique.

De 1936 à 1939, les Américains Howard Aiken et George Stibitz développent un calculateur automatique utilisant des relais électromécaniques.

En 1944, Eckert et Mauchly créent l'ENIAC, qui compte 18 000 tubes à vide, moins puissant qu'une calculette d'aujourd'hui.

Mais la véritable naissance de l'informatique se place en 1946, à Princeton aux Etats-Unis, quand le mathématicien **Von Neuman** définit une

machine qui porte son nom et que l'on appelle aujourd'hui l'**ORDINATEUR** (programme chargé).

Depuis cet événement, plusieurs générations d'ordinateurs ont été construites.

Première génération

En 1950 : commercialisation de l'ENIAC 1 toujours proposé par Eckert et Maucly. C'est une machine fonctionnant avec des tubes à vide.

Deuxième génération

En 1960 : Les tubes à vide sont remplacés par des transistors à semi-conducteurs, puis par des circuits intégrés comportant de plus en plus de transistors.

Troisième génération

Les premiers circuits sur silicium commencent à être produits au début des années 1970.

C'est à cette date que l'on voit apparaître les premières calculatrices de poche et les montres à affichage — diodes électroluminescentes, constituées d'un nombre restreint de circuits intégrés.

C'est surtout la création de super-ordinateurs où la capacité et la puissance deviennent considérables.

Les coûts de fabrication des mémoires et des processeurs se divisent par 10 000. C'est grâce à cette baisse vertigineuse des prix que l'ordinateur individuel peut voir le jour. Il ne coûte alors que quelques milliers de francs.

La quatrième génération

Elle se prépare actuellement dans les laboratoires de recherche. Ce sont les cryogéniques-ordinateurs. Ils utilisent les principes de la supraconductivité des circuits électrononiques, abaissés à des températures voisines du zéro absolu (– 270°). La consommation électrique de ces machines est pratiquement nulle. Les vitesses de calcul sont multipliées par dix.

1.2 VOIES DE DÉVELOPPEMENT

On distingue aujourd'hui trois voies de développement.

LA MÉGA-INFORMATIQUE

Caractérisée par d'énormes ordinateurs.

STAR ONE (CDC), le plus rapide : 250 millions d'instructions à la seconde, CYBER 250, 3084 IBM, UNIVAC 1130, etc...

Ils sont très coûteux et très complexes.

Les développements sont longs à mettre au point, et demandent une très grande qualification professionnelle informatique.

LA MINI-INFORMATIQUE

Ordinateurs de taille raisonnable et d'un prix plus accessible

PDP 11, IBM 38, etc...

Les développements sont plus aisés. Il suffit d'un informaticien pour faire les programmes. Celui-ci doit être, malgré tout, un très bon technicien et surtout un homme polyvalent.

LA MICRO-INFORMATIQUE

Apparue en 1970, avec les ordinateurs construits autour d'un seul microprocesseur

ORIC-1, TO 7, APPLE II, ZX 81 ETC...

Leur prix est faible et à la portée de toutes les bourses : environ le prix d'un téléviseur couleur.

Ils sont simples d'emploi et peuvent être utilisés par tous, sans compétences particulières.

1.3 DOMAINES DE L'INFORMATIQUE

Le mot français « **ORDINATEUR** » a été inventé en 1955, suivi par le mot « **INFORMATIQUE** » en 1962. L'informatique est une véritable

révolution industrielle qui est en train de pénétrer tous les secteurs de l'activité humaine.

1.3.1 Informatique scientifique

Suivi des vols spatiaux

C'est l'un des facteurs qui ont le plus contribué au développement de l'ordinateur. Deux raisons majeures permettent effectivement d'expliquer ce phénomène.

1. Nécessité de calculer très rapidement et avec une très grande précision des modèles mathématiques de plan de vols, calcul de trajectoires etc. la rapidité et l'exactitude des résultats contribueront pour la plus grande part au succès des missions. Il ne faut pas oublier les vitesses très élevées et les distances gigantesques à parcourir. Une erreur de 1/1 000e de degré sur la trajectoire Terre-Lune donnera à l'arrivée un écart de 300 km sur l'objectif à atteindre.

2. Nécessité d'autre part d'embarquer des ordinateurs pour une plus grande autonomie des engins. La légèreté est de rigueur. Il serait impensable de concevoir une navette spatiale avec un ENIAC à bord (ENIAC = 30 t). Il fallait donc réduire au maximum le poid et le volume des ordinateurs embarqués. Nous sommes habitués à ce que la réalité dépasse la fiction. Von Neuman pensait-il un jour que son calculateur ne pèserait que quelques milligrammes ?

Astronomie

La vitesse d'un ordinateur joue un rôle moins important que la puissance de calcul qu'il développe. A l'échelle des galaxies et des milliards d'années-lumière qui nous séparent d'elles, temps et distance semblent se rejoindre. L'astronome a trouvé un partenaire et un outil prodigieusement puissant pour l'accomplissement de ses travaux.

Physique nucléaire

Bien que les travaux aient été entrepris longtemps avant la naissance de l'ordinateur, c'est grâce à celui-ci — pour notre bonheur ou notre malheur — que des progrès spectaculaires ont pu être réalisés en matière d'armement et de nouvelle énergie. La fusion nucléaire est un processus d'une très grande complexité. Sans l'ordinateur, les calculs seraient presque

impossibles à faire à la main. Les centrales atomiques, les générateurs de courant électrique, en sont largement pourvues.

Climatologie

Voilà un domaine d'activité bien critiqué. C'est la Météorologie sous toutes ses formes. Mais rendons grâce aux Météorologistes. Le temps qu'il fera demain dépend d'une quantité de paramètres et de données innombrables. Les modèles de prévision météorologique programmés aujourd'hui sont encore imparfaits. Ils intègrent peu de données à l'échelle de notre globe terrestre. Ces données doivent ensuite être analysées, puis interprétées avant que la période donnée ne soit écoulée. Les renseignements obtenus sont diffusés et laissés à notre interprétation, nous, humbles mortels... qui oublions parfois notre parapluie dans le métro.

Il est aussi difficile de prévoir le temps qu'il fera, que de prévoir le déplacement d'un bouchon sur la Seine, de sa source à son embouchure. Si les calculs devaient être faits à la main, il faudrait que 50 000 personnes y travaillent durant 24 h, alors que l'ordinateur les donne en 2 h.

Laboratoire de recherche

Ce sont de très gros consommateurs d'ordinateurs. C'est certainement par là que les progrès les plus inespérés sont en cours d'aboutissement.

Des simulations très poussées sur des modèles génétiques programmés ont permis de découvrir des cellules nouvelles, des bactéries de toutes sortes, etc...

Ces modèles de simulation vont enfin supprimer le triste usage des cobayes. L'ordinateur est aussi à l'origine de récents progrès en matière de recherche pour explorer l'infiniment petit.

Dans les laboratoires de mathématiques, la visualisation de certains modèles de courbes permet de découvrir des images inattendues.

1.3.2 Informatique de gestion

C'est le domaine qui compte le plus d'ordinateurs. Vers les années 55, quelques grandes entreprises seulement en possédaient. Aujourd'hui, grâce à la miniaturisation des circuits et à leurs faibles prix, l'ordinateur devient accessible aux petites et moyennes entreprises, voire même aux professions libérales.

Les gestionnaires, comptables, statisticiens ont rapidement compris qu'en l'ordinateur ils trouveraient l'outil idéal pour l'accomplissement de leurs travaux.

La gestion des stocks est tenue en temps réel, les réapprovisionnements sont déclenchés automatiquement par le système.

La paye du personnel dans les grosses entreprises se fait en une journée, sans que l'on soit obligé de bloquer un service entier pour faire les calculs.

En comptabilité, les comptes sont tenus au jour le jour, les bilans, balances, journaux sont édités à la demande. Le chef d'entreprise peut ainsi avoir une meilleure observation de l'établissement qu'il dirige.

Les statistiques ont une place de choix dans l'ordinateur. Les chiffres y sont très dépouillés et s'accompagnent de graphiques de plus en plus précis.

1.3.3 Informatique industrielle

Bureaux d'étude, contrôle des machines-outils, robotique

L'informatique s'est implantée tardivement dans ce secteur. Mais le retard se comble rapidement avec la naissance des machines-outils pilotées par ordinateurs. Viennent ensuite la CAO (**C**onception **A**ssistée par **O**rdinateur) puis la CFAO (**C**onception et **F**abrication **A**ssistée par **O**rdinateur) qui bouleversent complètement les structures du bureau d'étude et des ateliers de fabrication. Les méthodes de conception (dessin, plan, etc.) sont entièrement changées par des équipements de plus en plus sophistiqués. Les progrès apportés sont considérables et sans précédent.

La robotique prend une place de plus en plus importante et pose dès à présent des problèmes humains. Mais n'allons-nous pas vers la société des loisirs ?...

Transports

Là aussi l'ordinateur a parfaitement trouvé sa place. Le contrôle aérien n'est aujourd'hui possible qu'assisté d'importants centres de calculs.

A bord des avions, l'informatique est de plus en plus utilisée. Les impressionnantes cabines de pilotage se simplifient. Les instruments de bord, voyants, aiguilles, etc... sont avantageusement remplacés par des écrans cathodiques. La définition des images et la précision des renseignements apportent un meilleur confort et une plus grande sécurité de pilotage.

Les défaillances techniques de l'appareil sont immédiatement décelées. L'équipage peut ainsi intervenir plus rapidement.

Les simulateurs de vol sont de merveilleuses machines reproduisant fidèlement la conduite de n'importe quel avion. Les images artificielles projetées devant le cockpit sont d'un réalisme étonnant. Les instruments de la cabine de pilotage y sont reproduits avec une grande exactitude et restituent, comme dans la réalité, toutes les indications de vol. Les constructeurs de ces machines ont poussé le raffinement jusqu'à reproduire les sensations d'accélération, trous d'air, freinage etc... Le plus fantastique est sans doute la possibilité de paramétrer toutes les configurations de vol (trajet, distance, météo, heure, charge, etc...), le tout sous le contrôle d'une batterie d'ordinateurs. Ainsi, les compagnies aériennes économisent jusqu'à 50 % d'heures de vols pour entraîner leurs élèves-pilotes. En outre, le simulateur pardonne sans dommage une faute grave de pilotage.

Les trains, métros, s'automatisent de plus en plus. Cela permet un trafic plus important, et surtout un gain de temps et une sécurité accrue. Ainsi les habitants de Lille ont un métro piloté et contrôlé entièrement par un système d'ordinateurs.

L'automobile n'est pas oubliée. Un certain constructeur y a même ajouté un digitaliseur de voix humaine pour prévenir le conducteur en cas d'incident. La circulation des grands centres urbains n'est pas oubliée. La circulation, les feux rouges, la régulation du trafic passent sous le contrôle de l'ordinateur.

1.3.4 L'informatique médicale

Surveillance des malades

Un domaine en plein essor. Grâce à un système de sondes et capteurs, le malade est relié à un laboratoire automatique contrôlé par un ordinateur. Les analyses sont faites en permanence, les résultats sont soumis à l'observation d'un système informatique qui, en fonction de seuils critiques, déclenchera une alerte au médecin de garde. Ainsi les soins peuvent être apportés immédiatement avant que le mal ne s'aggrave. De plus en plus pratiqué dans les centres hospitaliers, ce genre d'équipement rend d'importants services. Hélas, son prix est encore très élevé.

Le scanner, outil merveilleux, sonde notre corps sans provoquer de lésions. Sa conception n'est possible qu'assisté d'un ordinateur ayant une bonne résolution graphique. C'est une aide très efficace pour le diagnostic. Il évite bien des interventions exploratoires inutiles.

La médecine nucléaire, et autres procédés d'investigation de notre corps humain, est apparue avec la naissance de l'ordinateur. Leur résultat contribuera pour la plus large part au progrès de la médecine et de la recherche médicale.

1.3.5 L'éducation

Aide à l'enseignement, contrôle des connaissances

On a compris depuis longtemps que l'ordinateur serait un partenaire et un maître idéal pour l'enseignement. C'est un professeur infatigable et d'une patience illimitée. Il a l'avantage de ne jamais se mettre en colère contre l'entêtement de l'élève. Des recherches aux Etats-Unis ont démontré que le processus d'assimilation pourrait être multiplié par six. Les logiciels, les programmes, sont encore assez rares. Ce domaine de l'informatique prendra un essor réel quand la volonté du corps enseignant sera majoritairement positive. Nous comprenons bien les réticences. Les systèmes sont encore trop complexes à mettre en œuvre, et les langages pas très bien adaptés à cette fonction.

En France, l'IREM (**I**nstitut de **R**echerche pour l'**E**nseignement des **M**athématiques) travaille activement à la conception de didacticiels de mathématiques.

1.3.6 Informatique de bureau

Encore appelée la BUREAUTIQUE

C'est la machine à écrire à mémoire. C'est aussi la machine à traitement de textes. Ces machines s'implantent de plus en plus dans les secrétariats et services de dactylographie. Les utilisateurs, d'abord réticents devant l'aspect technique, sont aujourd'hui convaincus de l'efficacité et du gain de temps. Les coûts d'archivage des documents sont divisés par 100,

sachant qu'une disquette magnétique de traitement de textes peut contenir plus de 100 000 caractères.

1.3.7 Informatique personnelle

C'est vous, c'est nous, c'est tout le monde. Elle est aujourd'hui présente partout : au bureau, dans le train, chez soi, dans l'électroménager, dans notre montre, dans la télé, la chaîne HIFI, les petits jeux à affichage à cristaux liquides. Elle envahit littéralement notre univers quotidien et on ne s'en aperçoit presque pas, sinon qu'elle nous rend la vie plus agréable.

C'est aussi l'ordinateur individuel, les nuits blanches passées dessus, le BUG (*) introuvable dans le programme mais présent, là, devant nos yeux.

C'est aussi les clubs informatiques qui naissent un peu dans tous les coins. Ils sont très utiles et sont de véritables plateformes d'échange d'idées et de programmes et surtout des centres de formation « sur le tas » pour tous les débutants.

L'ordinateur individuel s'intègre pleinement dans la société des loisirs. Il y fait même une pénétration en force, voire inattendue. Il se taille la part du lion avec les jeux électroniques, vidéo, Wargames et jeux d'aventures.

(*) BUG veut dire CAFARD (insecte) en français. Il doit être pris comme synonyme d'erreur. L'usage de ce mot est très galvaudé dans la profession des informaticiens. L'origine vient de l'anecdote suivante : Au début de la conception des ordinateurs, chez un grand constructeur, un ordinateur refusait obstinément de fonctionner. Pourtant, ingénieurs, savants et techniciens avaient passé la machine au crible. Malgré tout, l'erreur persistait. Un soir, un balayeur zélé décida de dépoussiérer une boîte pleine de circuits électroniques. Quelle ne fut pas sa surprise en s'apercevant qu'à l'intérieur grouillait une famille de cafards, cause de la panne. La suite de l'histoire ne nous dit pas ce qu'il advint du balayeur...

CHAPITRE **2**

Le calculateur numérique ou l'ordinateur

2.1 QU'EST-CE QU'UN ORDINATEUR ?

Un ordinateur est un **MOYEN** et non une fin en soi.

C'est un **OUTIL,** dépourvu de toute imagination, qui ne peut rien sans l'être humain.

L'ordinateur est le prolongement du cerveau humain en matière de **RAPIDITE** mais pas en matière d'intelligence.

2.2 QUE TROUVE-T-ON DANS UN ORDINATEUR ?

LES MÉMOIRES

Ce sont des éléments de stockage. On les classe en deux groupes :

1. La mémoire centrale.
2. Les mémoires auxiliaires

L'UNITÉ CENTRALE

C'est le **MOTEUR** de l'ordinateur. Les Anglosaxons l'appellent **CPU** (**C**entral **P**rocessing **U**nit ou CPU).

Sur les mégas et minis ordinateurs, elle se compose de plusieurs microprocesseurs.

Sur les microordinateurs, un seul microprocesseur a la charge de toutes les fonctions.

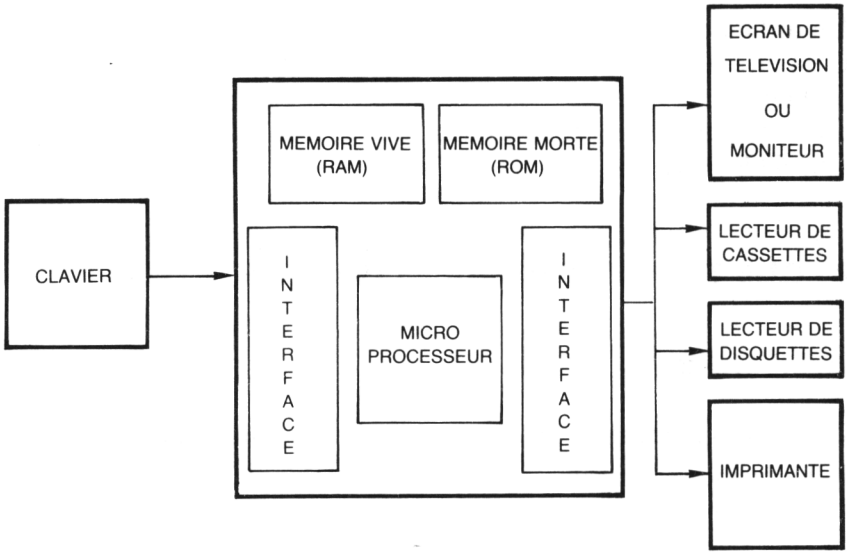
LES INTERFACES

C'est le réseau interne de circulation, pour véhiculer des données entre l'unité centrale, la mémoire, les périphériques.

LES ORGANES D'ENTRÉE/SORTIE

Ce sont les éléments de **COMMUNICATION** avec l'être humain.

STRUCTURE SCHÉMATIQUE D'UN ORDINATEUR

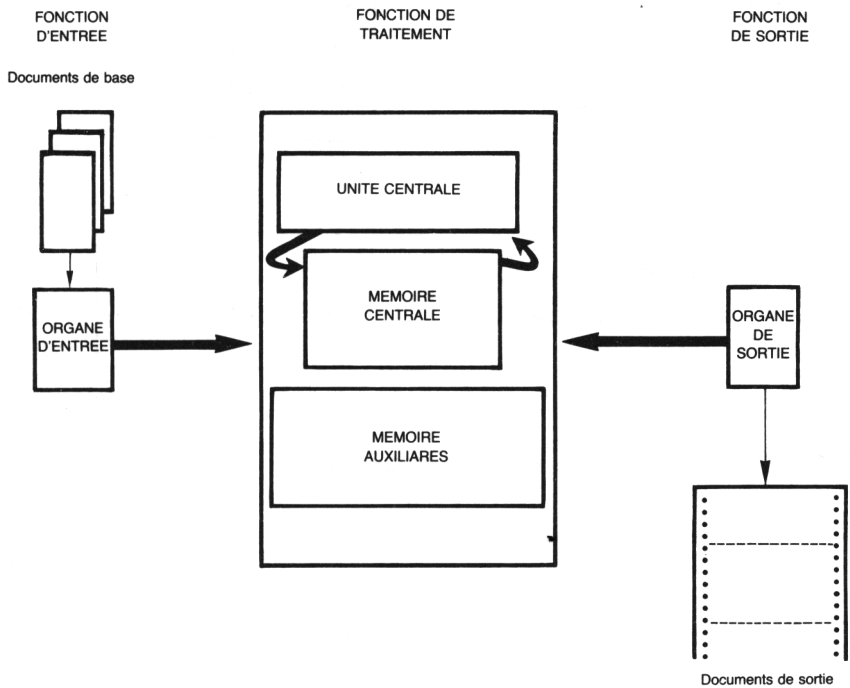


RAM = **R**ANDOM **A**CCES **M**EMORY : mémoire à accès aléatoire.
C'est la partie de mémoire prise par l'utilisateur.
On pourra y stocker des programmes ou des données, mais attention, cette mémoire est effacée par toute coupure de courant.

ROM = **R**EAD **O**NLY **M**EMORY : Mémoire à lecture seulement.
C'est dans cette mémoire morte que le langage BASIC est inscrit.
Elle est ineffaçable sur les coupures de courant, ou sur d'éventuelles écritures par erreur de manipulation.

2.3 FONCTIONS PRINCIPALES DE L'ORDINATEUR

L'objectif principal de l'ordinateur est de traiter des DONNEES.
L'homme, en interprétant les données ainsi traitées, dispose d'une masse importante d'informations.



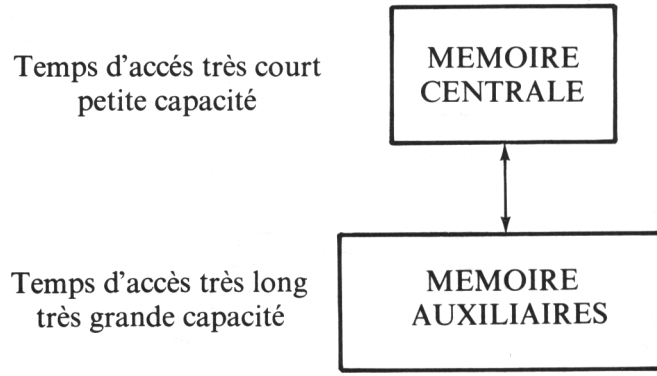
2.4 LES MÉMOIRES

Organe essentiel pour l'ordinateur comme pour le cerveau humain, il permet :

— D'**ENREGISTRER** des données
— de **CONSERVER** des données
— de **RESTITUER** des données.

Les deux rôles fondamentaux des mémoires sont :

— Le **TEMPS D'ACCES** C'est le temps nécessaire à la recherche et à la lecture des données.
— La **CAPACITE** C'est le nombre de caractères que l'on peut stocker.



2.4.1 La technologie des mémoires

MEMORISER c'est avoir la possibilité de restituer la donnée telle qu'elle a été enregistrée.

Pour mieux comprendre le fonctionnement d'une mémoire d'ordinateur, essayons de faire la comparaison avec la mémoire de notre cerveau.

MEMOIRE DE L'ETRE HUMAIN	MEMOIRE DE L'ORDINATEUR
Pourvue de la capacité du souvenir	Pourvue de la capacité de conserver des données
Elle enregistre des données communiquées par nos sens (vue, ouie, etc.)	Elle enregistre des données communiquées par des organes électro-mécaniques (claviers, lecteurs magnétiques etc.)
Le souvenir, c'est encore la possibilité de faire appel à une donnée enregistrée, que notre intelligence interprètera selon les besoins du moment et selon notre état d'âme.	Le souvenir n'existe pas en tant que tel, mais la mémoire est capable de restituer des données suivant la demande du traitement (programme). Etat d'âme totalement inexistant.

Nos souvenirs sont très nombreux et n'ont pas nécessairement de rapport entre eux. exemple : le parfum du jasmin et le théorème de Thalès	Même remarque : seul le traitement peut donner une relation d'ordre pour fournir des résultats.
Notre pensée, notre esprit, siège de l'intelligence est capable d'interprétations nouvelles et surtout d'adaptation à des situations imprévues.	L'ordinateur est totalement dépourvu de réflexion, d'intelligence. Il est incapable de s'adapter à une situation nouvelle non prévue par le programmeur.
Mécanisme de mémoire très mal connu chez l'homme.	Mécanisme parfaitement connu et maîtrisé puisque l'homme en est l'inventeur.

Le mythe selon lequel l'ordinateur est le maître de l'homme n'est que pure fiction. Ces craintes sont nées chez ceux qui ignoraient sa technologie et son fonctionnement.

Observons des mécanismes qui ont la faculté de mémoriser des données.

* la photographie est une mémoire → sels d'argent modifiés par la lumière
* un disque microsillon ..... → gravure sur matière plastique
* les livres ..... → dépôts d'encre sur papier
* la sculpture ..... → transformation de la pierre.

Ces supports de mémoire ne peuvent faire l'objet de modification sans détruire ce qu'ils représentent. Ils seront liés à leur unique fonction de mémorisation d'une donnée précise.

Si on veut sculpter un personnage sur une sculpture existante, la fonction première de mémorisation disparaît, le premier personnage sera détruit.

* un aimant ..... → orientation des cristaux de fer
* une ampoule électrique ..... → excitation d'un filament de carbone
* une bande magnétique ..... → orientation des particules de ferrite

* un transistor.....→ conductibilité d'un semi-conducteur sous l'influence d'un courant électrique
* l'eau.....→ modification de son état sous l'influence de la température.

Peuvent subir des modifications.

L'eau est un corps qui peut prendre trois états quand le paramètre température change.

Moins de 0 degrés .....→ solide
de 0 à 99 degrés .....→ liquide
plus de 99 degrés .....→ gazeux.

La transformation subie n'est pas irréversible.

Ce sont ces éléments qui ont intéressé l'homme pour construire des mémoires artificielles.

Continuons nos observations et aboutissons à une relation entre supports mémoire et représentation mathématique.

2.4.2 Que contient la mémoire

Une ampoule électrique servira notre démonstration.

Cet objet très familier ne peut prendre que deux états distincts sous l'influence d'un courant électrique.

LE COURANT PASSE	**: L'AMPOULE ECLAIRE**
LE COURANT NE PASSE PAS	**: L'AMPOULE N'ECLAIRE PAS**

Si nous associons à ces deux états une **valeur mathématique** nous obtiendrons un système **BINAIRE.** Boole et Leibnitz en ont donné les définitions.

Valeur 1 → LE COURANT PASSE → L'AMPOULE ECLAIRE
Valeur 0 → LE COURANT NE PASSE PAS → L'AMPOULE N'ECLAIRE PAS

Avec deux ampoules électriques qui ont des commandes distinctes nous obtiendrons les quatre combinaisons suivantes :

AMPOULE A	AMPOULE B	VALEURS BINAIRES		ETAT
		ampoule A	ampoule B	
pas de courant	pas de courant	0	0	0
pas de courant	courant passe	0	1	1
courant passe	pas de courant	1	0	2
courant passe	courant passe	1	1	3

Dans une mémoire d'ordinateur, l'ampoule est remplacée par des petits éléments électroniques que les Américains ont nommés **BIT**. C'est la plus petite unité de mémoire.

Comme l'ampoule électrique, chaque BIT prendra deux états possibles sous l'influence d'un courant électrique. Il est possible de connaître l'état de chaque bit de la mémoire.

Ces bits sont regroupés le plus généralement par **8** pour former un **OCTET** ou BYTE en anglais. C'est l'unité de capacité de la mémoire.

L'unité de mesure des mémoires est 1 024 octets. On a l'habitude de dire K octets (*). Ainsi nous trouverons la plupart du temps sur les dépliants des constructeurs d'ordinateurs la capacité de la mémoire exprimée dans cette unité.

Un ordinateur de 48 k possédera : 48 × 1 024 × 8 bits de mémoire.

Un octet de mémoire peut être modifié ou lu. Pour cela la position qu'il occupe doit être connue. Cette position est appelée **ADRESSE MEMOIRE.**

Il y a donc une adresse par octet. C'est un numéro obtenu par calcul.

BIT : Contraction de Binary digit.
(*) Le **K** est la lettre qui désigne le KILO ou encore 1 000.
Pour les mémoires, le KILO octets sera égal à 1 024 octets (2^8 octets).

Avec deux ampoules, nous obtenons deux fois plus d'états qu'avec une seule

2 ampoules donneront 2×2 états ..... 4 états
3 ampoules donneront $2 \times 2 \times 2$..... 8 états
4 ampoules donneront $2 \times 2 \times 2 \times 2$ ..... 16 états
.....
.....
8 ampoules donneront $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 =$ 256 états

Ces résultats, 2, 4, 8, 16, ..., 256, sont des **puissances de deux.**

Nous pourrions faire la même démonstration en remplaçant les ampoules par des BITS.

Ce qui conduit à dire qu'un octet pourra prendre 256 états différents. Si nous associons une valeur arithmétique à chaque état, nous pourrons mémoriser des nombres de 0 à 255 par octet.

Il sera possible d'associer un caractère à une valeur comprise entre 0 et 255.

Pour cela, il existe des tableaux de correspondance : ce sont les codes ASCII ou EBCDIC que vous trouverez en annexe.

Pour la commodité de lecture, l'affichage du contenu de la mémoire ne se fera pas BIT à BIT mais OCTET par OCTET.

Deux modes d'affichage sont alors possibles :

L'affichage sous forme de valeur HEXADÉCIMALES donnant la valeur du caractère représenté dans l'octet.

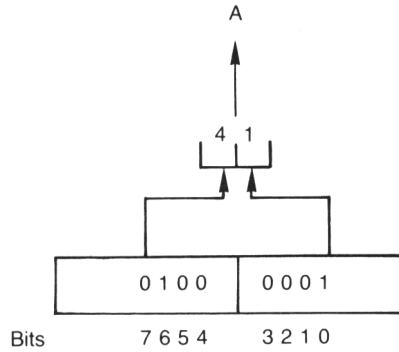
L'affichage directement sous la forme caractère.

Dans un microordinateur, la représentation des caractères passe par la norme internationale du code **ASCII**. Ainsi tout caractère occupera au maximum un octet.

Exemple : Comment le caractère **A** est représenté dans un octet ?

D'après le tableau du code ASCII ce caractère prend la valeur décimale **65.**

Cela donne comme valeur HEXADECIMAL **41.**



ATTENTION

Tout est fonction de l'interprétation qui sera faite sur la signification de la valeur contenue dans un octet.

L'ordinateur ne connaît que des nombres binaires.

Dans l'exemple précédent, 65 peut représenter soit un nombre, soit le caractère A.

Seul le CONTEXTE D'UTILISATION permettra de le dire.

Mais au fait, nous parlons BINAIRE, HEXADECIMAL, etc...

Connaissons-nous bien la définition de ces termes et savons-nous compter avec d'autres bases que le sacro-saint DECIMAL ?

Nous pourrions faire une petite révision sur les bases numériques les plus utilisées aujourd'hui, ensuite nous reprendrons l'étude de l'ordinateur.

Avertissement

Il n'est pas nécessaire de savoir compter en binaire ou en hexadécimal pour programmer. Les langages évolués suppriment cet inconvévient.

Mais si nous voulons approfondir l'étude de l'ordinateur, et programmer en langage machine, alors cela devient pratiquement indispensable.

2.4.3 Le binaire : quelques révisions

INITIATION AUX BASES NUMÉRIQUES

Qu'est-ce que le calcul décimal ?

C'est l'arrangement de **DIX SYMBOLES,** où la syntaxe et les règles d'écriture sont bien définies.

La liste des dix symboles est : **0, 1, 2, 3, 4, 5, 6, 7, 8, 9.**

On aurait très bien pu imaginer d'autres symboles qui auraient tout aussi bien fait l'affaire comme par exemple :

& " ' (§ ç , % *)

Ce qui donnerait la table de conversion suivante :

&	"	'	(	§	ç	,	%	*	)
0	1	2	3	4	5	6	7	8	9

Le nombre 768 539 s'écrirait alors : %,*ç()

Pas très lisible !

Règle d'écriture

Le symbole **0** ne représente dans le système décimal aucun objet. C'est encore le symbole signifiant l'absence d'élément.

Le symbole **1** représente **L'UNITE**. Ex. : Une pomme, un ordinateur...

. . . . . .

Le symbole **9** représente **NEUF** unités. Ex. : NEUF pommes, neuf ordinateurs.

Si l'on désire ajouter une unité de plus au neuf déjà existants, nous serions en mesure de chercher le symbole suivant après 9.

La liste étant épuisée, nous pourrions, en associant deux symboles, représenter symboliquement **NEUF + 1**

10 représentera **DIX** pommes, **DIX** ordinateurs...

Cette association représente donc **une unité de DIX objets.**

Et ainsi de suite. Nous pourrons énoncer symboliquement des quantités très grandes avec un minimum de symboles et des règles d'écriture très simples.

Introduisons maintenant le système binaire

Historique

Nos ancêtres — qui n'étaient pas nécessairement des Gaulois mais qui savaient compter — avaient certainement un système primitif pour faire du dénombrement.

Ils auraient pu utiliser des symboles de représentation comme : des cailloux, des traits dans le sable, des dessins sur la roche.

Pour ce qu'ils avaient à dénombrer, cette syntaxe était largement suffisante. Mais imaginez un instant que, de nos jours, il faille compter en unité-caillou ou unité-doigt !

HISTOIRE

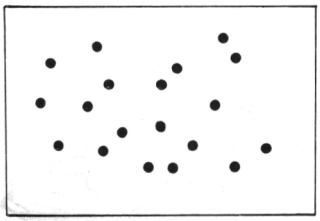
Il était donc une fois un petit berger des temps préhistoriques qui, soucieux de ne pas perdre de bêtes, désira dénombrer son troupeau.

Hélas : à cette époque-là, les gens ne savaient pas compter. Ils ne connaissaient pas le calcul décimal. Instruction et intelligence ne sont pas nécessairement liées.

Alors, le petit berger eut l'idée de regrouper ses moutons par deux, puis chaque groupe de deux encore par deux et ainsi de suite jusqu'à ce qu'il ne puisse plus faire de regroupement.

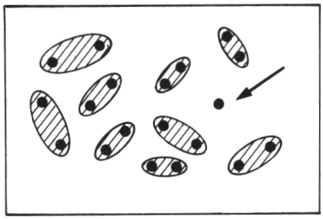
Il n'oublia pas à chaque étape de dessiner le symbole O, quand il ne restait pas de groupe isolé, et le symbole | quand un mouton ou un groupe de moutons ne pouvait être associé à un autre identique.

Voilà les règles qu'il s'est imposé et puis voici le champ de moutons :

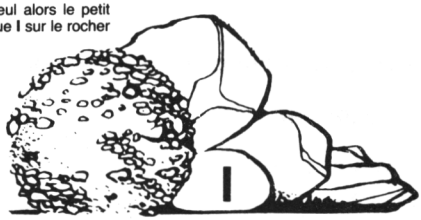


Un beau champ bien de chez nous avec dix neuf moutons.

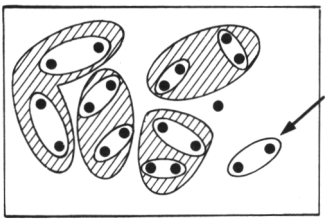
Le petit berger procéda au premier regroupement par deux. Ce qui donna :



Un mouton resta seul alors le petit berger fit une marque **I** sur le rocher

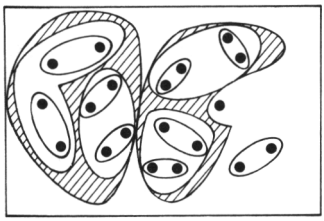


Puis il procéda au deuxième regroupement.

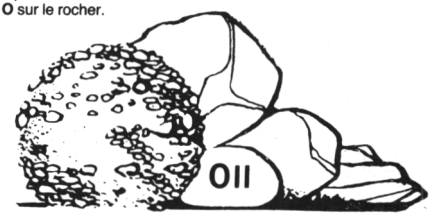


Il trouva un groupe de deux qu'il ne put associer à un autre groupe de deux. Il fit une deuxième marque **I** sur le rocher.

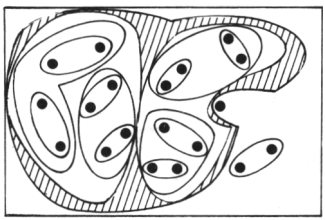
Il procéda ensuite au troisième regroupement.



Cette fois, plus de groupe seul. Alors il dessina le symbole **O** sur le rocher.

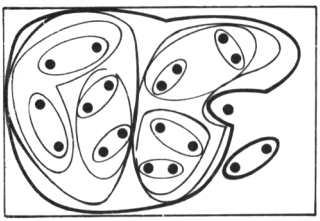


Il regroupa encore, pour la quatrième fois.

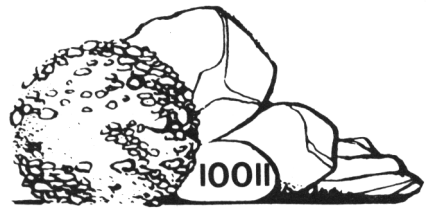


Même constatation qu'au coup précédent.
A nouveau il dessina le symbole **O**

Enfin, le petit berger essaya de faire un regroupement par 32, ce qui lui était impossible. Il restait donc un groupe de 16 moutons.



Il fit alors une dernière marque I sur le rocher.



Pour résumer, il obtint un mouton seul, puis un groupe de 2 seul, 0 groupe de 4 et de 8 seul, 1 groupe de 16 seul.

En additionnant le nombre de groupes successifs obtenus nous retrouvons l'égalité suivante :

$$\mathbf{10011_2 = 19} \quad \text{ce qui est aussi égal à} \quad \mathbf{1 + 2 + 16}.$$

Notre petit berger fut satisfait de son résultat, car il put s'apercevoir qu'il ne perdait pas de moutons, en comparant son résultat à celui de la veille.

Notre histoire s'arrête là. Mais au fait, comptait-il réellement ainsi et savait-il qu'il utilisait le système binaire ?

Il existe dans la nature une infinité de bases de calcul. Ces autres systèmes de dénombrement ont une symbolique qui leur appartient, mais les règles d'arrangement sont identiques quelle que soit la base de calcul utilisée.

Le système à base 2 est représenté par les symboles suivants :

0	1

Le système à base 16 est représenté par les SEIZE symboles suivants :

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Vous constaterez que les lettres A à F ont été rajoutées pour compléter la liste des symboles du système décimal.

Ce système de numération est appelé le système **HEXADECIMAL**.

Nous pourrions ainsi inventer des systèmes à bases 34 ou 258 : pourquoi pas ?

Commençons un tableau de correspondance entre les trois systèmes suivants.

Décimal	Binaire	Hexadécimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14
..	.....	..

Revenons à nos moutons !!!

2.5 L'UNITÉ CENTRALE

Cet organe est destiné au TRAITEMENT.

C'est le **MICROPROCESSEUR :** il date de moins de 10 ans.

Définition

Un microprocesseur est avant tout un circuit intégré, mais arrivé à un certain stade d'évolution. On peut dire que c'est un composant électronique, auquel les progrès de la technologie ont permis d'atteindre un niveau de complexité tel, qu'il est devenu un système informatique programmable d'usage général. On peut dire encore que les microprocesseurs sont des composants électroniques mettant à profit le remplacement du **cablé** par du **programmé** qui est la base de l'informatique.

Il se présente physiquement sous la forme d'une **PUCE** de silicium, portant des circuits intégrés de quelques fractions de centimètres carrés, montée dans un boîtier de céramique ou de plastique, muni de pattes ou de broches permettant la connexion avec les autres constituants de l'ordinateur. Ils sont à l'origine de la seconde révolution informatique.

Extrait de l'Encyclopédie de l'Ingénieur :
« Introduction aux micro-processeurs ».

L'unité centrale est construite avec un ou plusieurs microprocesseurs. Le plus généralement, sur les gros ordinateurs, elle se compose de :

1. L'UNITÉ DE COMMANDE

Elle décode l'ordre à exécuter qui se trouve en mémoire :

Mémoire centrale		
.....		
ordre	Position i − 1	Ordre déjà traité.
ORDRE	Position i	Ordre en cours.
ordre	Position i + 1	Ordre à traiter.
.....etc.		

a) Elle reconnaît à la position pointée que le code est bien un ordre.
b) Puis elle commande à l'unité arithmétique l'exécution de cet ordre.
c) Enfin elle passe à l'adresse i + 1 pour décoder le suivant.

2. L'UNITÉ ARITHMÉTIQUE ET LOGIQUE

Cet élément est le plus rapide de l'ordinateur. (Une addition est effectuée en un millionième de seconde environ.) Il a pour mission d'exécuter l'ordre (logique, arithmétique, comparaison, etc...).

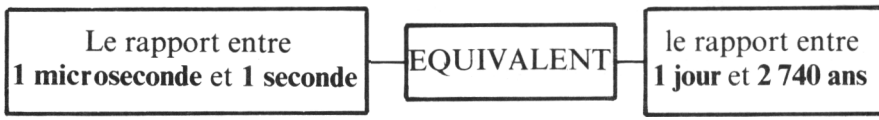
Là aussi n'oublions pas la notion de contexte. Rappelons-nous que la mémoire contient des valeurs pour chacun de ses octets.

Ces suites de valeurs peuvent être des enchaînements d'ordres ou bien des zones de données.

Donc, l'unité centrale doit être renseignée sur l'endroit où commence la zone qui contiendra le « programme » ou « suite d'ordres ».

2.6 TEMPS DE TRAITEMENT

Pour imaginer la rapidité d'un ordinateur, il faut garder à l'esprit l'équivalence suivante :



1 microseconde = un millionième d'une seconde.

Comparaison des temps nécessaires pour traiter 1 caractère

1. UNITE DE TRAITEMENT	0,1 microseconde (100 milliardième de seconde)	3 heures
2. MEMOIRE CENTRALE	1 microseconde	1 jour
3. LECTEUR DE DISQUETTES	20 microsecondes	20 jours
4. LECTEUR DE CASSETTES	2 500 microsecondes	7 ans
5. CLAVIER DE SAISIE	0,5 seconde	1 370 ans

2.7 LES ORGANES D'ENTRÉE/SORTIE

Ils sont appelés le plus souvent des **PERIPHERIQUES.**

Ils ont essentiellement pour rôle d'établir une **RELATION** entre l'ordinateur central et des **ELEMENTS EXTERIEURS.**

Ces éléments sont nombreux et diversifiés.

La finalité d'un ordinateur, comme toute autre machine, est de se mettre au service de **L'HOMME.** Il est donc naturel que les premiers moyens de communication soient orientés dans ce sens.

Les périphériques qui serviront ce dialogue sont :

LE CLAVIER DE LA MACHINE A ECRIRE
L'ECRAN DE VISUALISATION
L'IMPRIMANTE

On distingue aussi d'autres matériels périphériques. Ils ont des buts divers :

Les **MEMOIRES PERIPHERIQUES** permettent :

de mémoriser des données hors de la mémoire centrale,
de stocker les données en très grande quantité à des coûts très bas,
de transporter les données d'un ordinateur à un autre ordinateur,
de réduire les coûts de la mémoire centrale.

Les communications s'établiront dans le sens : ORDINATEUR → MEMOIRES PERIPHERIQUES.

On distingue plusieurs sortes de mémoires périphériques :

Les mémoires à support cartonné (Antiquité de l'informatique)

CARTE PERFOREE OU RUBAN PERFORE

Les mémoires à support ferromagnétique (l'actualité)

BANDE MAGNETIQUE
DATA CASSETTE
CARTE MAGNETIQUE
DISQUE MAGNETIQUE DUR
DISQUE MAGNETIQUE SOUPLE
CARTOUCHE MAGNETIQUE

Les mémoires à semi-conducteur (l'actualité)

MEMOIRE A BULLES MAGNETIQUES
MEMOIRE, PROM, EPROM, RPROM

Les mémoires à support optique (l'avenir)

DISQUE VIDEO (LASER)
à l'état d'étude

Les télécommunications ou la téléinformatique

Elles permettent de véhiculer des données sur de très grandes distances, et par des voies diverses.

Elles utilisent généralement les **RESEAUX DE COMMUNICATION DU TELEPHONE :**

Plus rarement les **ONDES HERTZIENNES**
De plus en plus les **CABLES OPTIQUES**
De plus en plus des **SATELLITES**

On utilise dans la plupart des cas, une boîte de connexion aux réseaux téléphoniques appelée **MODEM** (Modulateur/Démodulateur).

Les télécommunications servent à relier :

Un ordinateur à un autre ordinateur
Un ordinateur à un terminal intelligent
Un ordinateur à un terminal banalisé

La machine-outil

Depuis quelques années, elle est considérée comme périphérique si elle possède une interface de commandes numériques.

Elle est alors reliée à l'ordinateur central qui pilote les travaux à effectuer.

C'est encore la **ROBOTIQUE** et le domaine de la **CFAO.**

SONDES et CAPTEURS

Ils sont aussi des terminaux. Ils sont destinés à faire un travail très spécifique, propre à la fonction déterminée par le milieu ambiant.

Les résultats de leurs travaux sont envoyés à l'ordinateur central sous forme de signaux analogiques.

On distingue :

LES SONDES MEDICALES.
LES APPAREILS D'ANALYSES pour laboratoires de toutes sortes.
LES APPAREILS DE DETECTION.

Il manque à cette liste de nombreux éléments, mais l'essentiel est de montrer que les terminaux sont, et seront, de plus en plus présents dans notre vie.

C'EST LE PROLONGEMENT DE L'ORDINATEUR

2.8 LE SOFTWARE

INTRODUCTION

L'ensemble des éléments matériels, décrit dans le chapitre précédent, forme un tout qu'il est courant d'appeler **HARDWARE.**

Mot anglais qui veut dire :

HARD = DUR
WARE = MARCHANDISE

Par opposition à ce mot, le **SOFTWARE,** qui veut dire littéralement marchandise « molle », désignera l'ensemble des programmes, des données, des codes, etc...

Cet ensemble immatériel se nomme en français le **LOGICIEL.**

Comme les matériels constituant l'ordinateur, le rôle du logiciel sera capital pour l'accomplissement de tous les travaux à exécuter.

Il est indispensable de distinguer deux sortes de logiciel

1. **Les logiciels de base.** Ils ont pour rôle essentiel de gérer la machine.
2. **Les logiciels d'application.** Ce sont nos programmes ou des programmes déjà écrits et vendus dans le commerce.

Nous vous proposons donc d'étudier la première partie des logiciels dans ce chapitre.

Un ordinateur est constitué de composants électroniques de tout ordre, de périphériques de toute sorte, l'ensemble travaillant grâce à l'énergie électrique.

Afin que l'outil soit complet et qu'il réponde bien aux services attendus, les constructeurs d'ordinateurs ont réalisé des **programmes particuliers** livrés avec la machine.

L'ensemble de ces programmes joue un rôle très important pour la conception et l'exploitation des applications que les utilisateurs écriront.

On distingue deux fonctions essentielles que rempliront ces programmes :

1. Une fonction d'aide aux développements et tests des applications.
2. Une fonction d'assistance à l'exploitation, à l'exécution et la surveillance des matériels et de nos programmes.

Ces deux fonctions forment le **SYSTEME**.

2.8.1 Les outils d'aide au développement

Il n'est pas possible d'étudier, dans le présent ouvrage, toutes les techniques utilisées pour la conception de ces ensembles très complexes.

Citons les grandes lignes.

2.8.2 Les langages

Pour que deux individus puissent se comprendre, il est nécessaire qu'ils parlent le même langage.

C'est exactement la même chose entre l'homme et l'ordinateur.

Le langage doit être compris des deux. N'oublions pas qu'un ordinateur est construit avec des circuits électroniques. Ils ne comprennent qu'un seul langage : LE BINAIRE (1-0).

Grâce au microprocesseur (la puce) fabriqué à l'aide de ces circuits, un langage s'établit en s'appuyant sur une logique à base deux, et suivant les spécifications des constructeurs de microprocesseurs.

C'est le **CODE MACHINE**

Il y en a aujourd'hui de plusieurs sortes et de diverses provenances.

Les plus connus sont : 6502 (ORIC-1, Apple // ...)
Z80 (Sharp, ...)
6809 (TO7, ...)

On pourra écrire des programmes en code machine. C'est aussi et seulement ce code qui sera exécuté, que ce soit un micro-ordinateur ou un méga-ordinateur.

Et la programmation dans les langages évolués ?

C'est tout simplement une traduction, ou une interprétation des mots et des symboles du langage, faite par un **super-programme** écrit en code machine.

Ces traducteurs ou interprètes ont pour nom **COMPILATEUR ou INTERPRETEUR.**

Citons les plus connus :

Les langages de gestion : COBOL, PL1, etc...
Les langages scientifiques : FORTRAN, ALGOL, APL, etc...
Les langages spécifiques : LOGO, SIMULA, etc...
Les langages universels : PASCAL, BASIC, ADA, etc...

Afin que les programmeurs de tels outils puissent enregistrer leurs programmes en mémoire centrale ou en mémoire auxiliaire, des outils d'édition sont à leur disposition. Ils n'ont pas tous la même façon de fonctionner, mais ils ont tous la même utilité.

Ainsi vous pourrez à loisir : enregistrer, corriger, tester.

Ces outils font partie de la famille des **EDITEURS**.

Nous aurons à notre disposition bien d'autres outils pour l'aide au développement et à l'exploitation. Deux se distinguent par leur efficacité et leur complexité. Ce sont :

LES SYSTEMES DE FICHIERS et les **BASES DE DONNEES**.

Ils sont orientés vers le traitement et le stockage de grandes quantités de données.

2.8.3 Les outils d'exploitation

Ils sont destinés essentiellement à coordonner l'ensemble des programmes d'application.

Ils assurent leur bon fonctionnement et rendent la tâche des programmeurs plus facile.

Ils ont des rôles très divers :

- Surveiller le fonctionnement de l'ordinateur.
- Interpréter les commandes.
- Gérer les affichages, les impressions.
- Contrôler le fonctionnement des périphériques (imprimante, etc.).
- Etc.

Ces outils sont, dans la plus grande majorité des cas, paramétrables. Leur usage est simple. Les résultats, et les services rendus, toujours parfaits.

CHAPITRE **3**

Une approche de l'analyse

CONSEILS

Bien qu'il n'existe pas de règles universelles pour réaliser une analyse, certains principes de bon sens doivent êtres respectés.

Dans un premier temps vous pourrez ainsi mener à bien des projets d'informatique personnelle. Puis, avec une bonne pratique du raisonnement et une expérience plus approfondie, vous serez à même d'entreprendre des projets plus ambitieux et, pourquoi pas, de qualité professionnelle.

Quelles sont les méthodes existantes ?

Chercheurs, logiciens, informaticiens ont énoncé et mis au point des modèles d'analyse. Ces modèles tendent tous à définir une méthodologie de travail selon des structures très précises. Ils se nomment **WARNIER, CORIG,** et bien d'autres encore pour ne citer que les plus connus.

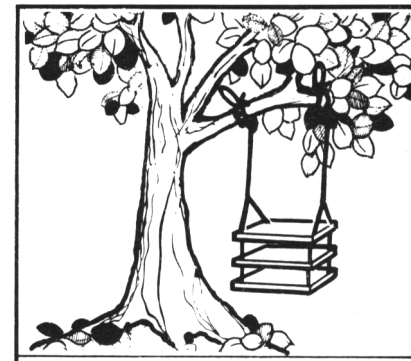
Vous n'êtes pas obligé, pour programmer, de les adopter mais ils ont l'indiscutable avantage d'exister. Ils guident l'analyste et le programmeur dans un environnement satisfaisant et très sécurisant.

Il faut absolument adopter une méthode, et éviter à tout prix de concevoir des projets dépourvus d'analyse qui n'ont ni queue ni tête.

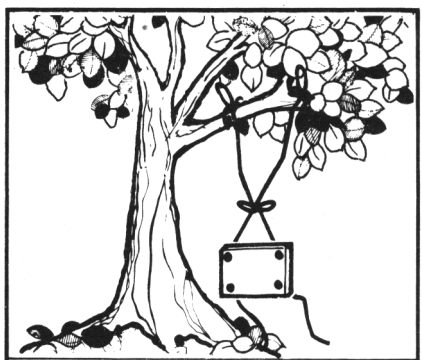
Se mettre devant l'ordinateur avec une vague idée, sans avoir défini les structures organiques et algorithmiques du travail à faire, vous conduira obligatoirement vers un résultat très médiocre ou qui ne fonctionnera pas dans tous les cas.

Il n'est pas rare de trouver, même chez les professionnels, des cas de figure ressemblant au croquis suivant :

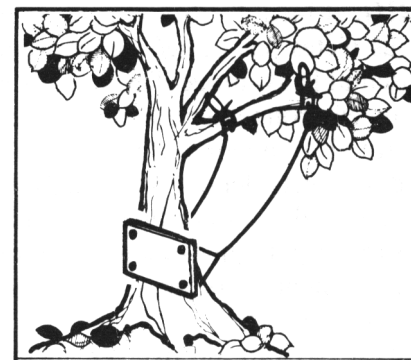
Ce que demande le client



Ce que l'analyste a compris



Ce que le programmeur a fait



Après corrections apportées

Ce qu'il était suffisant de faire

CONDUITE D'UNE ANALYSE

Une analyse doit toujours être conduite du cas général vers le cas particulier.

C'est la méthode dite **DESCENDANTE.** C'est-à-dire du haut vers le bas. Les Anglosaxons l'appellent **TOP-DOWN.**

Effectivement, cette méthode fait appel à la logique naturelle des choses. Dans la vie quotidienne, nous appréhendons d'abord les problèmes et les événements sous l'aspect global, puis nous découpons partie par partie ce que nous voulons mieux connaître, ou ce que nous devons détailler.

Nous ferons exactement la même chose pour la conduite d'un projet informatique. Vous devrez respecter cette règle fondamentale sous risque de voir conduire vos travaux à de cuisants échecs, sinon à une débauche d'efforts, de mises au point, et de tests inutiles. Il vaut mieux vérifier le bon fonctionnement d'un programme que de tester ses erreurs.

Avant de programmer, il est indispensable de posséder, parfaitement bien, l'analyse fonctionnelle du modèle à réaliser. Il est impensable de laisser des points dans l'incertitude, voire dans l'ignorance totale.

Les cas généraux sont toujours simples à résoudre. Par contre, il est délicat de traiter les cas particuliers. Ils sont souvent litigieux et mal compris du futur utilisateur du programme.

N'essayez pas d'analyser des algorithmes fleuves. Découpez en parties simples, et réalisez complètement chaque partie, avant de passer à la suivante. Ce conseil s'applique à la programmation. Il est préférable de programmer beaucoup de petits programmes simples qu'un gros très compliqué.

Enfin, un point nous semble important : il faut bien connaître l'ordinateur sur lequel l'application s'exécutera. Ceci vous évitera bien des erreurs de conception organique.

Ces préceptes sont très généraux et seront repris dans le paragraphe intitulé **LA PROGRAMMATION STRUCTUREE.**

CHAPITRE **4**

Introduction à la programmation

4.1 INTRODUCTION À LA PROGRAMMATION

PROGRAMMER.....c'est quoi au juste ?

— C'est donner des **INSTRUCTIONS,** des **ORDRES** que l'on fera exécuter plus tard par l'ordinateur.
— En général, les ordres sont **SIMPLES.**
— Ils sont **ENREGISTRES** dans la mémoire de l'ordinateur.
— On peut les **MODIFIER** quand on veut.
— On peut même **TOUT CHANGER** aussi !

4.2 LA PROGRAMMATION À LA PORTÉE DE TOUS

INTRODUCTION

Cette partie est plus particulièrement destinée aux personnes qui n'auraient aucune idée des concepts de la programmation et de l'informatique.

En s'aidant d'exercices simples, voire même évidents, nous vous dirigeons vers une autre façon de réfléchir aux problèmes et solutions que vous rencontrez dans ce domaine si passionnant.

Enfin cette méthode vous conduira en douceur vers la programmation en **BASIC.**

Il n'est pas nécessaire d'être mathématicien, physicien ou électronicien pour programmer.

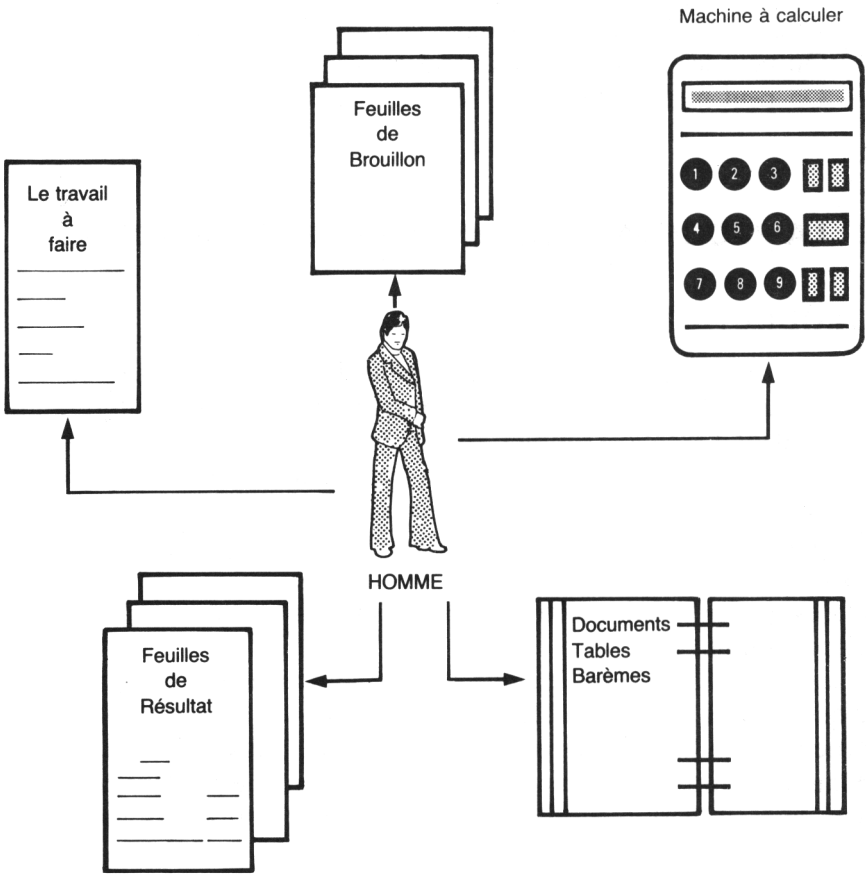
Il faut simplement être logique et faire le petit effort d'apprendre les quelques règles fondamentales de l'informatique, ainsi que le jeu des instructions du langage de programmation que vous utiliserez.

Mais attention, l'ordinateur est une machine qui ne pardonnera aucune erreur de votre part, bien que réalisant des calculs très complexes et très rapides.

Les trois premiers exercices vous demandent de vous mettre à la place de l'ordinateur. Vous serez en quelque sorte l'acteur d'une pièce dont nous sommes les scénaristes, ceci dans le seul but de vous faire comprendre les mécanismes internes de fonctionnement.

Les exercices suivants vous démontreront comment, à partir d'un problème posé, nous aboutissons à un programme écrit en basic. Ceci vous permettra ensuite de devenir scénariste, c'est-à-dire auteur de programmes que vous aurez conçus, imaginés.

4.3 LE CALCULATEUR HUMAIN



Quelles sont les **FONCTIONS** réalisées dans le déroulement des opérations ?

* **LES FONCTIONS DE CALCUL**

— Réalisées par la calculatrice.

* **LES FONCTIONS DE CONSERVATION**

— Feuilles de résultats	Mémorisation définitive
— Feuilles de brouillon	Mémorisation temporaire
— Documents, tables, etc.	Fichiers, Banque de données, etc.

* **LES FONCTIONS DE SAISIE DES INFORMATIONS ET D'ECRITURE DES RESULTATS**

— Les manipulations sur la calculatrice.
— L'usage de crayons, stylos, gommes, etc...

* **LES FONCTIONS D'ENCHAINEMENT DES TRAVAUX**

— L'HOMME, aidé de fiches-travail à faire.

Vous saurez programmer quand vous aurez pris l'habitude de vous mettre à la place de l'ordinateur pour rédiger vos programmes.

4.4 EXERCICE N° 1 : AMUSEZ-VOUS À ÊTRE L'ORDINATEUR

Calculez, à l'aide de votre calculatrice, la somme des 10 nombres de la liste A

LISTE A	123	2	457	300	1	23	78	64	1000	67

a) Prenez votre calculatrice.
Vous êtes maintenant L'UNITE CENTRALE.
La calculatrice sera L'UNITE ARITHMETIQUE.

b) Exécutez dans l'ordre des séquences 10 à 90 le programme suivant.

10 DEBUT
20 APPUYER sur la touche MARCHE
30 RENTRER dans la calculatrice le premier nombre de gauche non rayé de la liste A, puis le RAYER de la liste A
40 APPUYER sur la touche M+
50 SI les nombres de la liste A sont tous rayés ALORS ALLER à la séquence 60
SI il reste des nombres dans la liste A ALORS ALLER à la séquence 30
60 APPUYER sur la touche RM
70 LIRE la réponse affichée sur L'ECRAN de la calculatrice
80 APPUYER sur la touche ARRET
90 FIN

Résultat du programme = 2115.

La touche **M+** *permet d'additionner une valeur affichée à la valeur contenue dans la case mémoire de la calculette.*

La touche **RM** *permet d'afficher la valeur contenue dans la case mémoire de la calculette.*

L'exercice que vous venez de réaliser n'apporte rien en tant que résultat mais il vous a montré schématiquement comment un programme est exécuté par un ordinateur.

Les séquences 10 à 90 représentent les étapes d'un programme écrit dans le langage « FRANCAIS ».

L'ALGORITHME, c'est-à-dire, l'ordre logique des séquences peut se représenter sous la forme d'un **ORGANIGRAMME.**

4.5 COMMENT FAIRE UN ORGANIGRAMME ?

1. L'organigramme schématise l'enchaînement logique des phases d'une analyse.

2. Il sera une aide efficace pour résoudre des problèmes complexes.

3. Il aura **toujours** un **DEBUT** et une **FIN.**

Organigramme de l'exercice 1 (fig. 1)

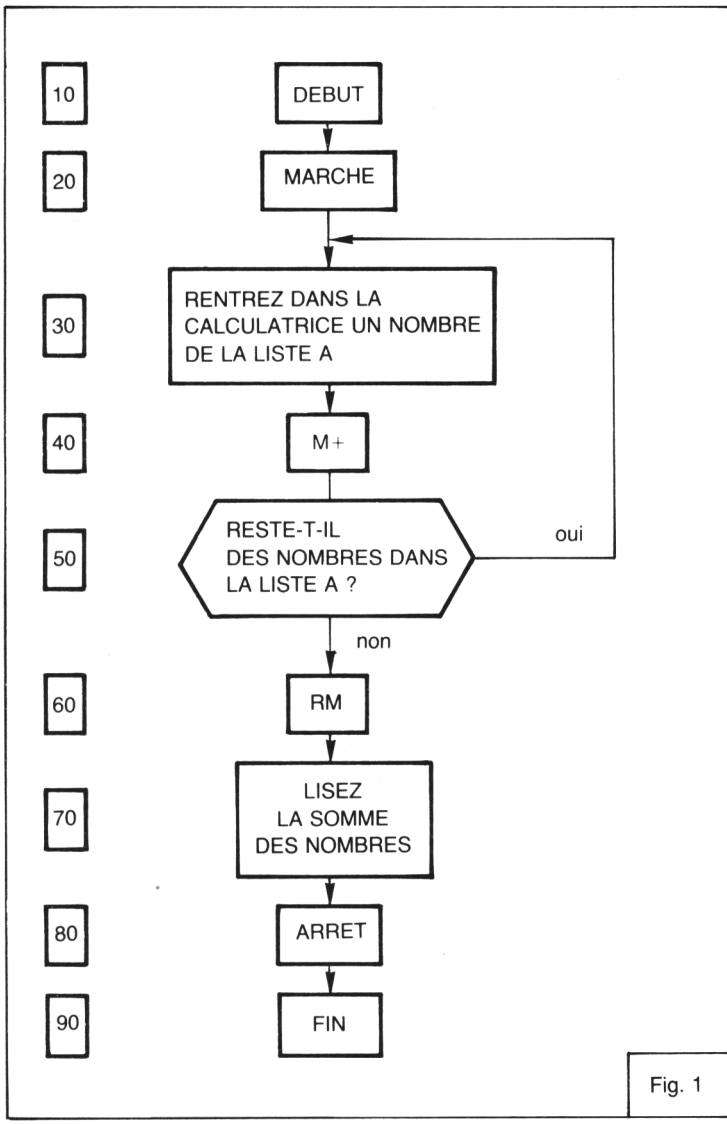


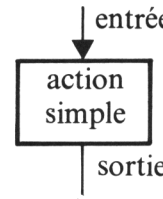
Fig. 1

4. Les expressions logiques de l'analyse seront abrégées et placées dans des ENCADREMENTS rectangulaires ou en losanges.

Chaque encadrement ou pavé sera relié au moins à un autre par des traits de raccordement.

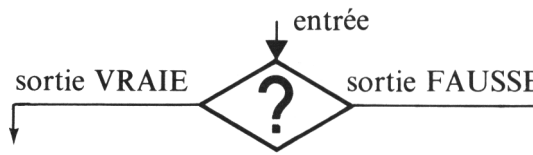
5. Les traits de raccordement seront orientés par des flèches indiquant le sens de la lecture.

6. Les expressions simples, ou sans condition d'enchaînement, seront placées dans des encadrés rectangulaires.

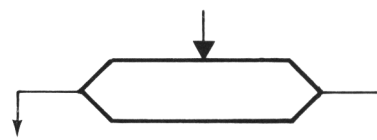


7. Il y aura une seule entrée et une seule sortie pour les encadrés rectangulaires.

8. Les expressions conditionnelles seront placées dans des encadrés en losange

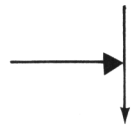


Pour des commodités de lecture, et par manque de place, le losange est réduit ainsi :



9. Il y aura une seule entrée et obligatoirement deux sorties pour les encadrés en losange.

10. Les traits de raccordement pourront se rejoindre pour former un regroupement.



Bien d'autres symboles existent. Vous les trouverez en annexe.
Expliquons les différents termes employés.

4.5.1 Qu'est-ce qu'une expression simple ?

C'est un ordre qui n'entraîne pas de condition.

Ex. : [Je travaille] est une expression simple

[+ 1 sur A] est une autre expression simple

Organigramme de deux expressions simples qui se suivent (Fig. 2).

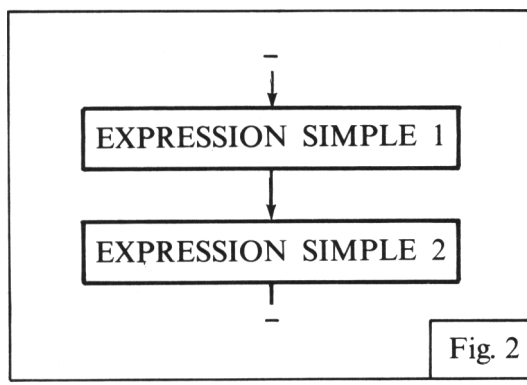
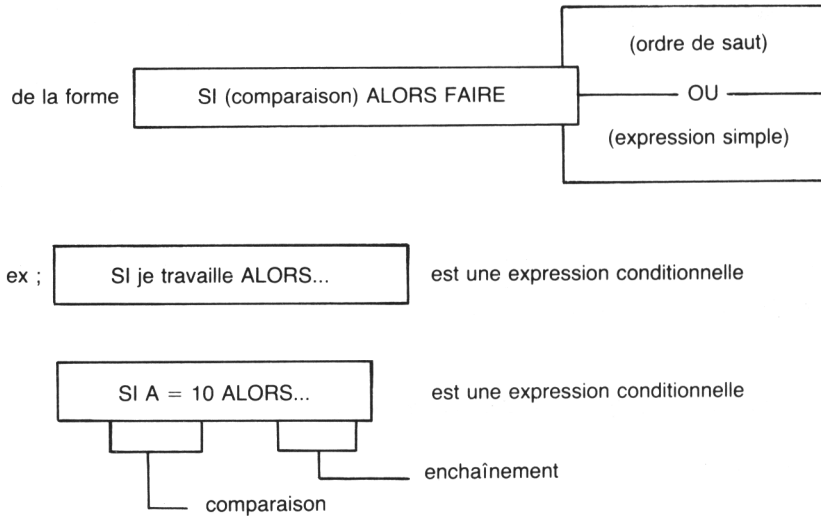


Fig. 2

4.5.2 Qu'est-ce qu'une expression conditionnelle ?

C'est une expression de **COMPARAISON et** de commande **D'ENCHAINEMENT.**



Partie d'organigramme avec une expression de comparaison (Fig. 3).

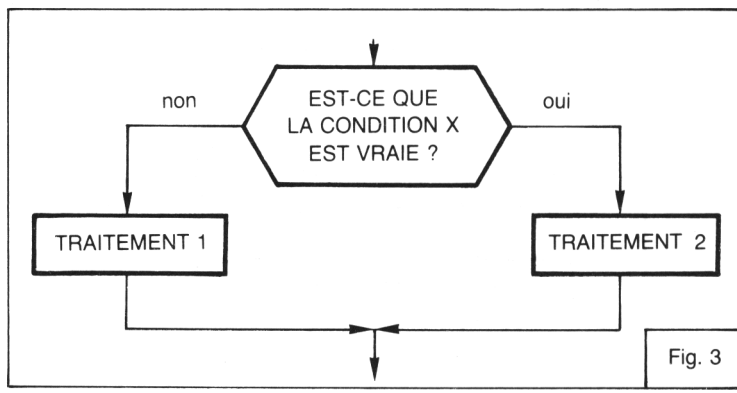


Fig. 3

Une autre forme d'une partie d'organigramme avec expression conditionnelle (Fig. 4).

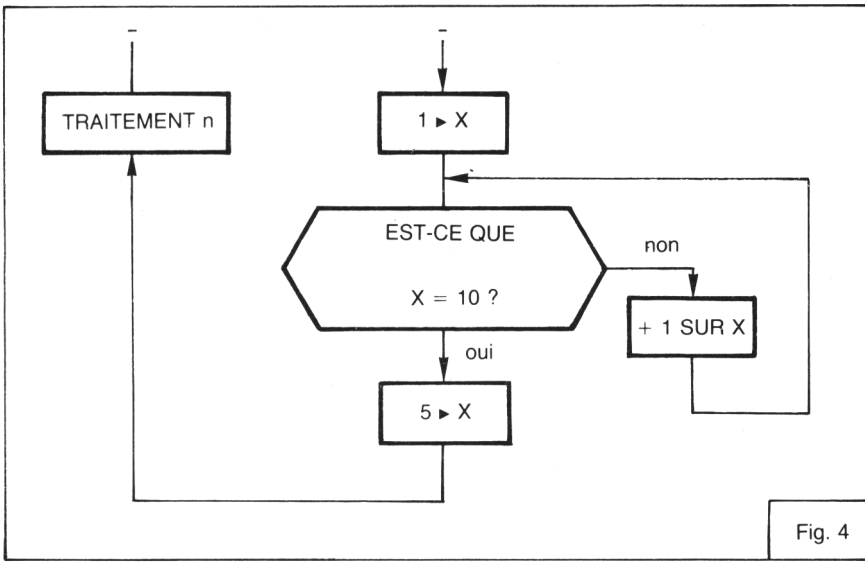


Fig. 4

FORMULATION DES EXPRESSIONS CONDITIONNELLES

Une expression conditionnelle peut être formulée de plusieurs façons.

4.5.2.1 TANT QUE (expression simple) FAIRE (expression simple)

L'expression **TANT QUE** sera représentée de la façon suivante (Fig. 5).

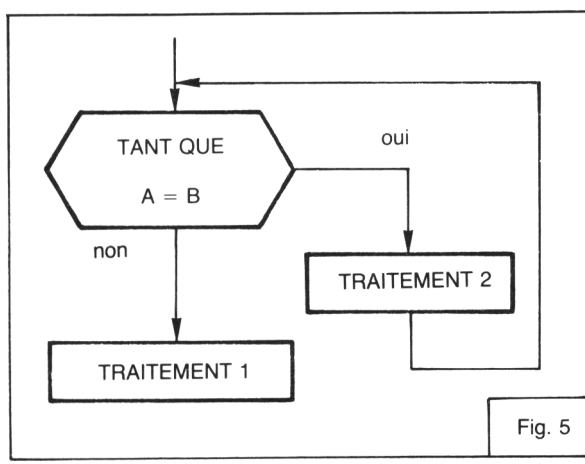


Fig. 5

4.5.2.2 SI (expression simple) ALORS (expression simple)

L'expression **SI** sera représentée comme suit (Fig. 6).

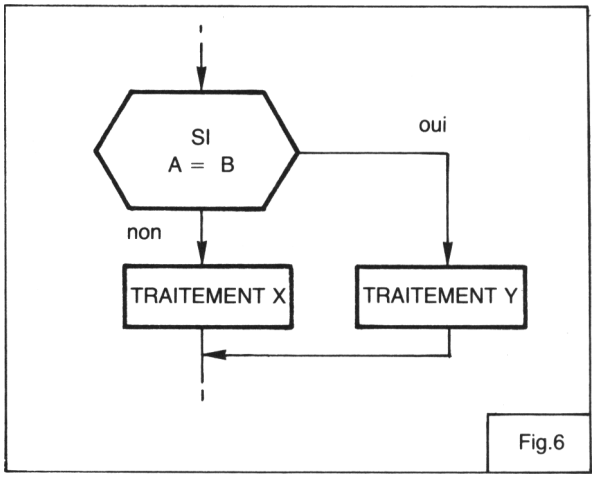


Fig.6

Il existe des expressions conditionnelles plus complexes. Elles sont caractérisées par une condition d'arrêt, quand un seuil de valeur est atteint.

4.5.2.3 POUR (x) COMMENÇANT À (i) ET ALLANT JUSQU'A (n) ALORS...

Qui sera représentée :

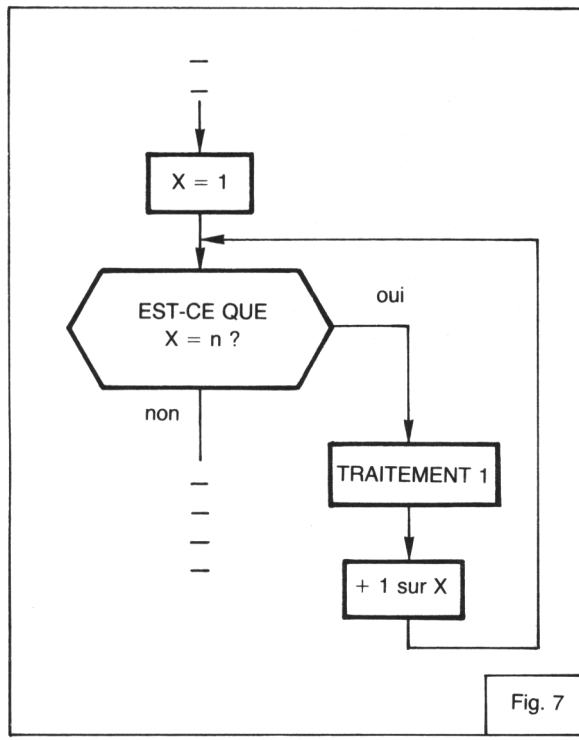


Fig. 7

4.6 EXERCICE N° 2

Cet exercice reprend l'exemple de l'exercice 1 mais ATTENTION : nous avons introduit dans les séquences du programme une expression conditionnelle supplémentaire.

LISTE A	132	2	457	300	1	23	78	64	1000	67

10 DEBUT
20 APPUYER sur la touche marche
30 RENTRER dans la calculatrice le premier nombre de gauche non rayé de la liste A, puis le RAYER de la liste A
40 SI le nombre entré est égal à 300, ALORS ALLER à la séquence 50.
SI NON, ALORS ALLER à la séquence 70
50 APPUYER sur la touche M –
60 ALLER à la séquence 80
70 APPUYER sur la touche M +
80 SI les nombres de la liste A sont tous rayés, ALORS ALLER à la séquence 90.
SI NON, ALORS ALLER à la séquence 30
90 APPUYER sur la touche RM
100 LIRE la réponse affichée à l'écran de la calculatrice
110 APPUYER sur la touche ARRET
120 FIN
RESULTAT DU PROGRAMME = 1524.

L'exercice 2 vous a présenté deux formes d'expression conditionnelle :

Première expression : EST-CE QUE LE NOMBRE EST EGAL A 300 ?

De la forme | **SI** nombre = 300 **ALORS...** |

Deuxième expression : RESTE-T-IL DES NOMBRES NON RAYES DANS LA LISTE A ?

De la forme | **TANT QUE** nombre non rayé dans liste A **ALORS...** |

QUELLES CONCLUSIONS POUVONS-NOUS TIRER DE CES PREMIERS EXERCICES ?

1. Un programme est **UNE SUITE ORDONNEE** d'ordres référencés par un numéro de séquence.

Ces ordres sont des **INSTRUCTIONS.**

L'ordinateur exécutera les instructions qui lui sont proposées, dans l'ordre où elles ont été référencées, en tenant compte des ordres de branchement aux séquences indiquées.

2. Il exécutera strictement ce que vous lui avez ordonné de faire.

3. Il n'aura pas la possibilité de deviner vos oublis, mais il détectera les erreurs ou impossibilités de fonctionnement que vous aurez programmées.

4. Les points 1, 2, 3 resteront toujours vrais quel que soit le langage de programmation utilisé. Seule la syntaxe des instructions sera différente.

4.6.1 Les aiguillages ou drapeaux

Plus généralement appelés **FLAGS**

A quoi servent-ils ?

A savoir si l'exécution d'un programme est passée par un point particulier de séquences.

Ils permettent de créer des **STRUCTURES ALTERNATIVES** de séquences, et de réaliser **UNE FOIS** un groupe spécifique de séquences.

Comment les construire ?

Une variable	+	un test sur la valeur de cette variable

Représentation d'un aiguillage, sous la forme d'un organigramme, qui définit une structure alternative entre TRAITEMENT 1 et 2 (Fig. 8).

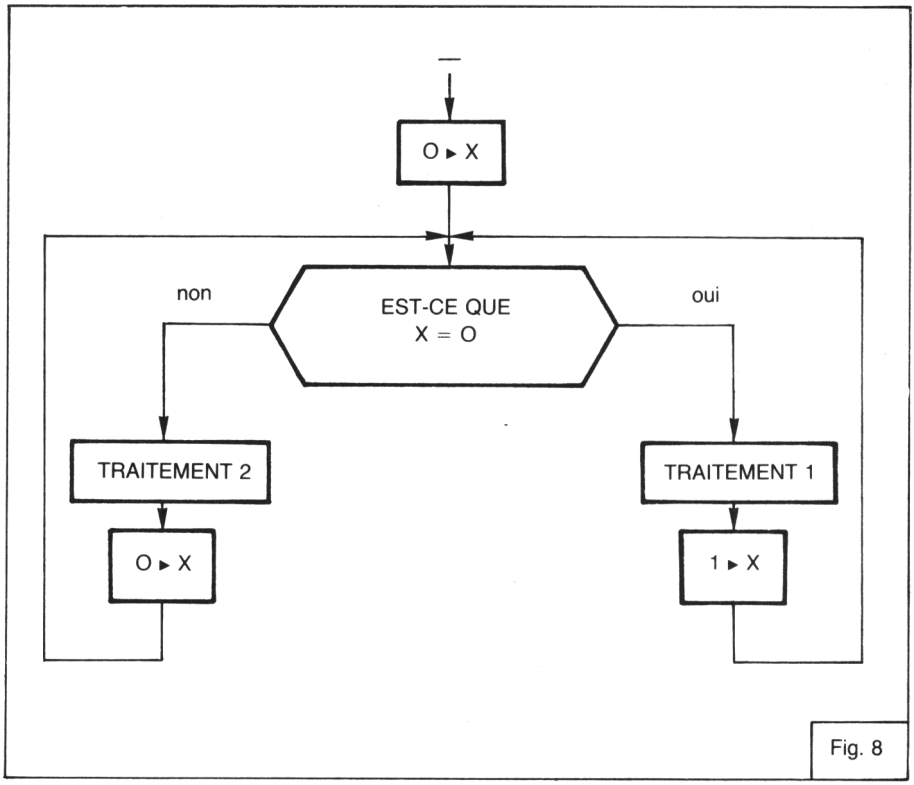


Fig. 8

4.7 EXERCICE N° 3 : LE TRI

Il est quelque peu différent des deux précédents car vous n'allez plus utiliser votre calculatrice.

Si vous voulez atteindre le résultat souhaité, il faudra redoubler d'attention quant à l'ordre des séquences du programme proposé.

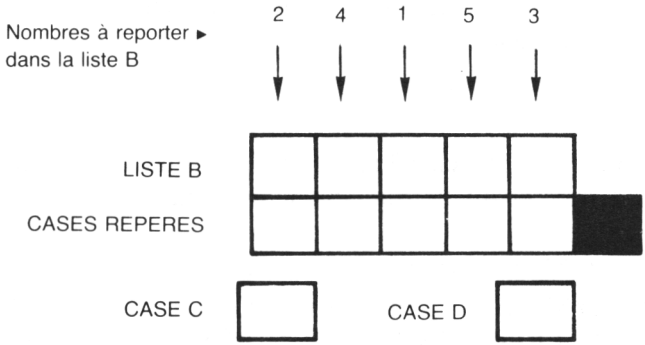
Enoncé du problème

On se propose de trier par ordre croissant les livres d'une bibliothèque, du plus petit au plus grand et de gauche à droite. Chaque dos de livre comporte un numéro.

Si manuellement l'opération est simple, par programme elle devient beaucoup plus délicate. Imaginez de faire ce travail de reclassement avec une seule main, sachant que vous ne pouvez avoir plus d'un livre à la main, et que le support intermédiaire (tablette de travail) ne peut soutenir plus d'un livre à la fois.

a) Prenez un crayon et une gomme
b) Vous êtes toujours l'unité centrale

La bibliothèque est représentée par les cases de la liste B.
La tablette de travail est représentée par la case C.



Programme de tri d'une liste de nombres (exercice 3).

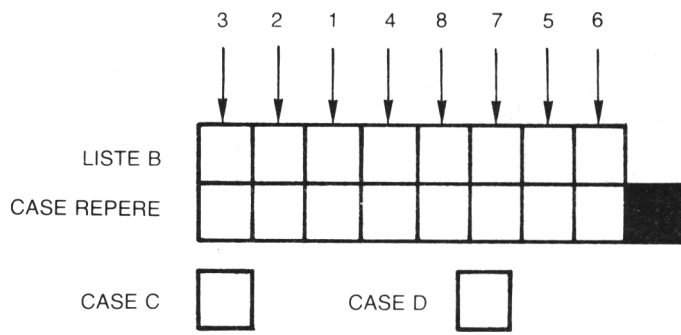
10 DEBUT
20 DESSINER une flèche dans la case repère située sous le premier nombre de gauche de la liste B
30 ECRIRE 0 dans la case D
40 SI la case repère suivante est noire ALORS ALLER à la séquence 140
50 SI le nombre de la case repérée par la flèche est plus petit que le nombre de la case suivante ALORS ALLER à la séquence 120
60 EFFACER le nombre de la case D, ECRIRE 1 dans la case D
70 ECRIRE dans la case C le nombre repéré par la flèche
80 EFFACER de la liste B le nombre repéré par la flèche
90 ECRIRE dans la liste B, à la case repérée par la flèche, le nombre situé dans la case suivante de la liste B

100 EFFACER de la liste B le nombre de la case suivante repérée par la flèche
110 ECRIRE dans la liste B, à la case suivante repérée par la flèche, le nombre situé dans la case C
120 EFFACER la flèche de la case repère, puis DESSINER LA FLECHE dans la case repère suivante
130 ALLER à la séquence 40
140 SI le nombre de la case D est égal à 0 ALORS ALLER à la séquence 180
150 SI toutes les cases repère sont noires, SAUF la première, ALORS ALLER à la séquence 180
160 NOIRCIR la case repère où se trouve la flèche
170 ALLER à la séquence 20
180 FIN

Le programme de l'exercice 3 restera valable quelle que soit la quantité de nombres à trier et quelle que soit leur valeur

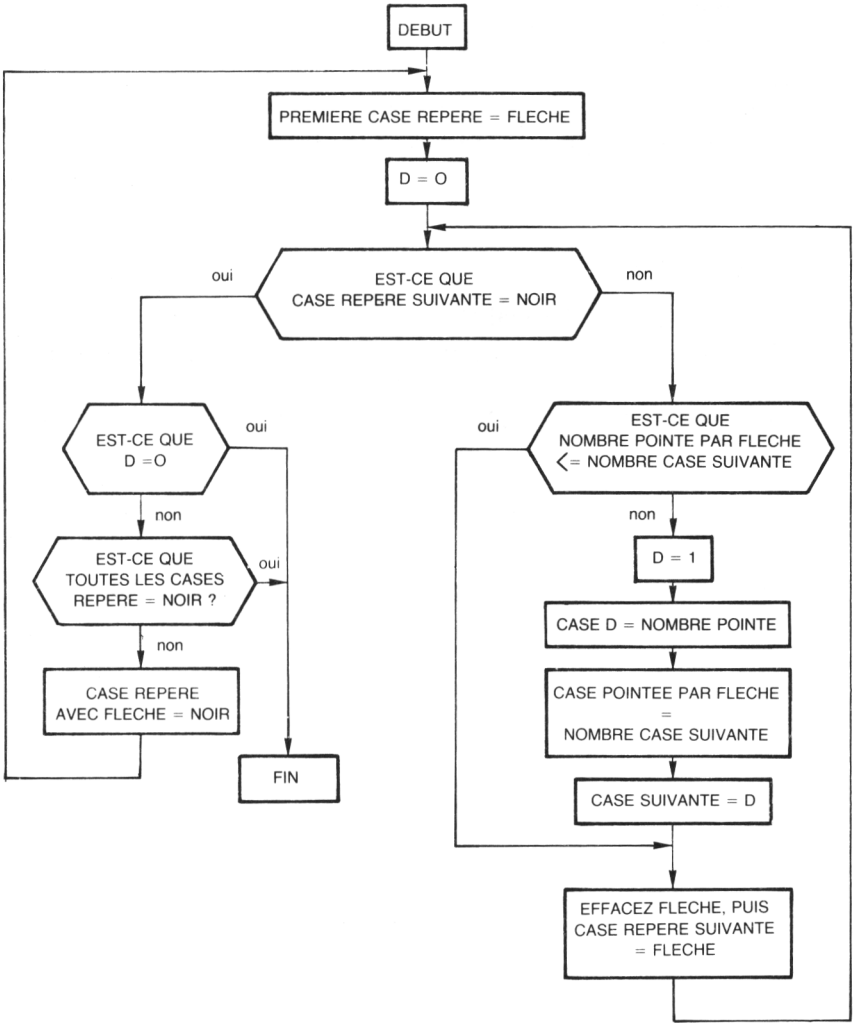
UNE ANALYSE BIEN PENSEE CONDUIRA NECESSAIREMENT A UN PROGRAMME BIEN ECRIT, FONCTIONNANT DANS TOUS LES CAS

Afin de vérifier cette règle amusez-vous à le faire exécuter sur la liste suivante.



Pour se rapprocher de la réalité d'écriture des programmes en langage machine (évolué ou non) nous formulerons de manière plus **CODIFIEE** les expressions des séquences 10 à 180 de l'exercice 3.

Organigramme de l'exercice 3



Cet exercice utilise la case D comme FLAG.

En fonction de la valeur qu'il prend, ce FLAG permet d'indiquer le passage aux séquences 60 à 110. Nous sommes ainsi protégé d'un éventuel tri sur une liste naturellement ordonnée.

CONVENTION DES SYMBOLES UTILISES

<	plus petit que
>	plus grand que
=	égal
< =	plus petit ou égal
> =	plus grand ou égal
#	différent

Ex. : $\boxed{A < B}$ A plus petit que B

LES AFFECTATIONS

1 → D	la case D reçoit 1	D = 1
+ 1 SUR D	on ajoute 1 au contenu de la case D	D = D + 1

ATTENTION : le contenu précédent est perdu sur une affectation.

CHAPITRE **5**

De la théorie à la pratique

5.1 AVANT-PROPOS

— En programmation, il ne suffit pas d'être inventif, ingénieux et astucieux. Il faut surtout savoir se discipliner et se contrôler.

— Dans toute activité humaine, une génération atteint à peine la surface de ses possibilités.

— Il en sera de même en programmation.

— La prochaine génération de programmeurs sera beaucoup plus compétente que la première.

— Le programmeur sera capable d'un niveau de précision et de productivité insoupçonné aujourd'hui.

— Cette nouvelle génération devra acquérir **CONTROLE et DISCIPLINE,** en apprenant à écrire des programmes correctement dès le début.

— Programmer est un travail logique et sérieux, qui demande **CONCENTRATION et PRECISION.**

H. D. MILLS
Federal Systems Division **IBM**

5.2 UN BON CONSEIL

Mettons en pratique les connaissances théoriques que vous avez acquises ou que vous allez acquérir.

RIEN NE PEUT REMPLACER LA PRATIQUE

Ces exercices vous ont certainement semblé très simples, mais ô combien fastidieux et longs. Les résultats attendus sont justes.

Jouer à être l'ordinateur c'est bien, mais quand on en possède un c'est mieux.

Nous vous proposons maintenant de déléguer ces tâches répétitives à votre microordinateur. Nous vous montrerons comment il est possible, d'une formulation écrite en français, de passer au langage **BASIC.** Enfin, nous en profiterons pour voir de quoi sont capables ces petites machines.

5.3 RAPPEL

HISTORIQUE

BASIC a été initialement développé à **DARMOUTH COLLEGE** par **JOHN KEMENY** et **THOMAS KURTZ,** vers les années 60.

La simplicité de ce langage a attiré l'attention des utilisateurs qui l'ont adopté rapidement.

Vous pourrez l'utiliser pour résoudre de nombreux problèmes courants de mathématiques, développer des modèles de physique, faire également de la gestion et de l'informatique personnelle, ainsi que la plus grande partie des applications possibles sur ordinateur.

Le langage BASIC (Beginner's All purpose Symbolic Instruction Code) est un langage de programmation simple, auquel s'ajoutent des mots d'anglais courants.

Il est constitué essentiellement :

d'**INSTRUCTIONS**
de **COMMANDES**
de **FONCTIONS**

ORIENTATION DU BASIC

ACCESSIBLE	à un non-informaticien
RELATION	conversationnelle Les instructions sont exécutables immédiatement. La mise au point des programmes est rapide car on peut disposer à tout moment de la liste des instructions et la modifier (avantage des langages interprétés)
COMMANDES	Je veux que la machine fasse tout de suite...
INSTRUCTIONS	Je prépare des opérations que je fais exécuter plus tard
E/S FICHIER	Mes données peuvent être ailleurs que dans la machine

E/S PERIPHERIQUE Je veux choisir l'endroit origine/destination.
(ECRAN, IMPRIMANTE, DISQUE, etc...)
E/S = ENTREE/SORTIE

MODES DE FONCTIONNEMENT

Il y a donc deux modes de fonctionnement sur un microordinateur.

1. Le mode **DIRECT**

L'exécution de l'ordre (commandes ou instructions) est immédiate. C'est aussi utiliser le microordinateur comme d'une calculette.

Ex. : PRINT SIN(45.13)
ou PRINT 98765/(3765.987*342.78)

2. Le mode **PROGRAMME**

Tout est là. Nous venons d'aborder ce mode dans le chapitre précédent. Approfondissons la question et apprenons à programmer.

Vous trouverez en annexe un tableau résumant toutes les instructions du BASIC disponibles sur ORIC-1.

5.4 REPRENONS L'EXERCICE N° 1

Rappel

10 DEBUT
20 APPUYER sur la touche MARCHE
30 RENTRER dans la calculatrice le premier nombre de gauche non rayé de la liste A, puis le RAYER de la liste A
40 APPUYER sur la touche M+
50 SI les nombres de la liste A sont tous rayés ALORS ALLER à la séquence 60
SI il reste des nombres dans la liste A, ALORS ALLER à la séquence 30
60 APPUYER sur la touche RM
70 LIRE la réponse affichée sur l'ECRAN de la calculatrice
80 APPUYER sur la touche ARRET
90 FIN

LISTE A	123	2	457	300	1	23	78	64	1000	67

Ce qui est évident

Les séquences 10 et 20 sont ignorées dans la mesure où votre ordinateur est déjà sous tension. Par mesure de sécurité, entrons la commande **NEW.** Ce qui a pour effet de nettoyer la mémoire.

Ce qui est moins évident

Comment allons-nous entrer les nombres de la **liste A** dans l'ordinateur, et par quel moyen allons-nous lui faire comprendre que ces nombres sont bien ceux de la liste A et pas ceux d'une autre liste ?

Puisque cette liste est fixe et qu'elle ne comporte que 10 éléments, nous vous proposons d'utiliser l'instruction **DATA** qui permet de stocker des données statiques.

Ce qui donnera comme instruction :

10 DATA 123,2,457,300,1,23,78,64,1000,67

Nous savons d'autre part qu'une calculatrice possède une ou plusieurs cases mémoire. Il faut donc indiquer à notre programme sur quelle case de mémoire se feront les additions.

Prenons par exemple — et tout à fait arbitrairement — une case mémoire, ou plus exactement une variable BASIC nommée X, et initialisons-la à 0.

Ce qui donnera comme instruction :

20 X = 0

D'autre part, à la séquence 50 de l'exercice 1, il faut se poser la question de savoir si tous les nombres de la liste A sont rayés. Nous vous proposons donc de compter le nombre de fois où le programme fera une addition.

Puisqu'il s'agit de compter, il faut un compteur, donc une deuxième variable BASIC qui sera utilisée comme telle. Nous vous proposons — tout aussi arbitrairement — la variable CP qui sera initialisée à 0.

Ce qui donnera comme instruction :

25 CP = 0

A la séquence 30 de l'exercice 1, nous devions rentrer le premier nombre de gauche non rayé de la liste A. Il existe en BASIC une instruction qui lira ce nombre de gauche non rayé (pas encore lu).

Ce qui donnera comme instruction :

30 READ N

Quand l'ordinateur aura exécuté cette instruction, la variable N contiendra le nombre lu de la DATA.

A la séquence 40 de l'exercice 1, nous devons appuyer sur la touche M+ de la calculatrice. Cette opération a pour effet d'additionner, au nombre déjà existant de la mémoire, le nombre que nous venons de rentrer. Cette opération se fera très facilement.

Ce qui donnera comme instruction :

40 X = X + N

Qui doit se lire : AJOUTER AU CONTENU DE LA VARIABLE X LE CONTENU DE LA VARIABLE N ET LE RÉSULTAT OBTENU RANGE DANS LA VARIABLE X.

Attention ! : Ce n'est pas une formule mathématique.

Cette instruction d'addition étant programmée, il faudra compter à chaque fois que l'ordinateur l'exécutera. Nous avons préparé pour cela un petit compteur.

Ce qui donnera comme instruction :

45 CP = CP + 1

Que nous reste-t-il à faire maintenant ?

D'après ce qui est prévu à la séquence 50 de l'exercice 1, nous devons nous poser deux questions.

Ce qui donnera comme instructions :

50 IF CP < 10 THEN GOTO 30

Que se passera-t-il quand le programme se déroulera ?

Admettons que la variable CP soit égale à 4 :

l'ordinateur analysera l'instruction de la façon suivante **SI** (IF) le contenu de CP **est plus petit** (<) que **10 ALORS** (THEN) **aller à** (GOTO) l'instruction 30.

La condition sera **VRAIE**, le programme continuera en 30.

Admettons que la variable CP soit enfin égale à 10 :

La condition sera **FAUSSE**, le GOTO 30 ne sera pas effectué, le programme continuera à l'instruction suivante.

Maintenant que les dix additions sont effectuées, il faut afficher le résultat à l'écran.

Ce qui donnera comme instruction :

60 PRINT X

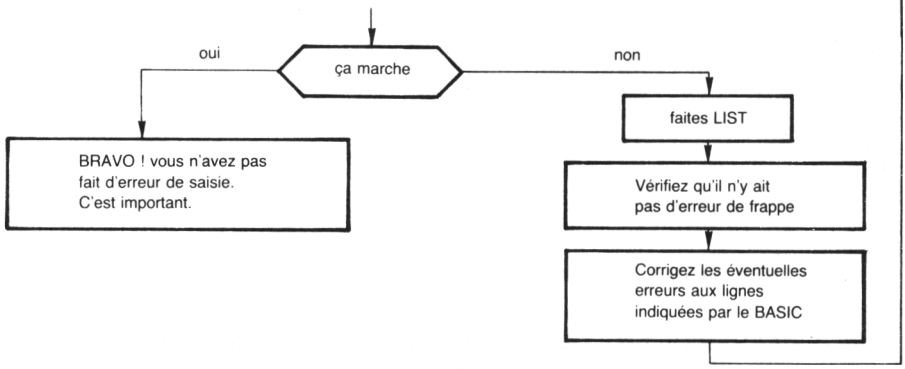
Les séquences 60 et 70 de l'exercice 1 sont inutiles. La séquence 90 se traduira par :

90 END QUI N'EST PAS OBLIGATOIRE EN BASIC

En résumé : Rassemblons toutes les instructions pour en faire un programme.

```
10 DATA 123,2,457,300,1,23,78,64,1000,67
20 X=0
25 CP=0
30 READ N
40 X=X+N
45 CP=CP+1
50 IF CP < 10 THEN GOTO30
60 PRINT X
90 END
```

Maintenant que votre programme est en mémoire de votre micro-ordinateur, vous pouvez à loisir, et autant de fois que vous le désirez, le faire exécuter. Pour cela entrer la commande RUN.



Nous espérons que ces quelques explications ont éveillé en vous une âme d'informaticien.

Vous venez certainement d'avoir le petit déclic qui vous a fait comprendre le concept de programme et la façon dont on procède.

Ce programme n'est intéressant que d'un point de vue **EDUCATIF.**

Nous vous proposons d'y apporter une petite modification qui vous permettra de faire les additions de dix nombres que vous entrerez au clavier de votre microordinateur.

— Il n'est plus nécessaire de conserver la liste des dix nombres.

Supprimons l'instruction DATA, en entrant au clavier le numéro de l'instruction qui la référence.

Frappez **10** sans rien derrière. Puis faites **RETURN**

Voilà qui est fait

Vous voulez vérifier que l'instruction a bien été détruite. C'est simple :

Frappez **LIST** ...

Vous voyez bien que l'ordinateur est à votre service !!!

— Les instructions **20** et **25** restent toujours bonnes.

— L'instruction **30** va être remplacée par une instruction de saisie au clavier.

Ce qui donne comme instruction :

30 INPUT N

Pourquoi N ? Pour dire tout simplement à l'ordinateur où doit être rangée la valeur saisie au clavier.

```
20 X=0
25 CP=0
30 INPUT N
40 X=X+N
45 CP=CP+1
50 IF CP < 10 THEN GOTO30
60 PRINT X
90 END
```

— Vous pouvez dès à présent rentrer cette nouvelle instruction, puis essayer votre nouveau programme en frappant **RUN.**

— Votre programme s'exécute... Vous apercevez sur votre écran un point d'interrogation. Cela signifie que le programme est en train d'exé-

cuter l'instruction 30 INPUT N et qu'il restera indéfiniment sur cette instruction si vous ne rentrez aucune valeur ponctuée par la touche **RETURN.**

Vous allez ainsi entrer dix valeurs différentes.

... ! Vous venez de voir apparaître sur l'écran un chiffre que vous n'avez pas tapé.

C'est le programme qui l'a affiché, et c'est le résultat des dix additions successives.

Vous pouvez, autant de fois que vous le désirez, rentrer la commande **RUN,** et autant de fois vous obtiendrez l'exécution du programme en mémoire.

Vous êtes encore un peu perplexe. Qu'à cela ne tienne, rentrez les nombres de la liste A et comparez le résultat de l'exercice 1 et celui obtenu par votre microordinateur...

Etes-vous convaincu ? Nous n'en doutons pas un instant.

5.5 PASSONS À L'EXERCICE N° 2

Ayons la même démarche que pour la programmation basic de l'exercice 1

Rappel

10 DEBUT
20 APPUYER sur la touche marche
30 RENTRER dans la calculatrice le premier nombre de gauche non rayé de la liste A, puis le RAYER de la liste A
40 SI le nombre rentré est égal à 300 ALORS ALLER à la séquence 50
SI NON ALLER à la séquence 70
50 APPUYER sur la touche M –
60 ALLER à la séquence 80
70 APPUYER sur la touche M +
SI les nombres de la liste A sont tous rayés ALORS ALLER à
80 la séquence 90
SI NON ALLER à la séquence 30
90 APPUYER sur la touche RM
100 LIRE la réponse affichée à l'écran de la calculatrice
110 APPUYER sur la touche arrêt
120 FIN

Ce qui ne change pas par rapport à l'exercice 1.

```
10 DATA 123, 2, 457, 300, 1, 23, 78, 64, 1000, 67
20 X = 0
25 CP = 0
30 READ N
```

Ce qu'il faut ajouter

La codification des tests de la séquence 80.

Ce qui donne comme instructions

```
40 IF N = 300 THEN GOTO 50
45 IF N < > 300 THEN GOTO 70
```

Cette programmation n'est peut-être pas très élégante mais ce qui nous intéresse plus particulièrement, c'est de comprendre. Après nous verrons comment améliorer le style.

A la séquence 50 de l'exercice 2, nous devons effectuer la soustraction du nombre lu avec le contenu de la variable X.

Ce qui donne comme instruction

```
50 X = X - N
```

— Cette instruction étant rentrée, nous devons ordonner à l'ordinateur d'aller à la séquence 80. Cela se traduira par le codage d'un ordre de saut inconditionnel à l'instruction 80.

Ce qui donne comme instruction

```
60 GOTO 80
```

— Il ne faut surtout pas oublier que dans le cas où le contenu de la variable N n'est pas égal à 300 nous devons programmer une addition.

Ce qui donne comme instruction

```
70 X = X + N
```

— Que l'on doive faire une addition ou une soustraction, il faut obligatoirement exécuter les instructions de condition d'arrêt.

Pour cela, nous allons tout simplement reprendre les instructions 45 à 90 du précédent programme et les numéroter différemment.

Ce qui donne comme instructions

```
80    CP = CP + 1
95    IF CP 10 THEN GOTO 30
100   PRINT X
110   END
```

— **Nous obtenons comme programme**

```
10    DATA 123,2,457,300,1,23,78,64,1000,67
20    X = 0
25    CP = 0
30    READ N
40    IF N = 300 THEN GOTO 50
45    IF N < > 300 THEN GOTO 70
50    X = X – N
60    GOTO 80
70    X = X + N
80    CP = CP + 1
95    IF CP < 10 THEN GOTO 30
100   PRINT X
110   END
```

— Vous pouvez l'exécuter en rentrant la commande **RUN**.
Le résultat est immédiat et juste.

5.6 EXERCICE N° 3

Rappel

10 DEBUT
20 DESSINER une flèche dans la case-repère située sous le premier nombre de gauche de la liste B
30 ECRIRE 0 dans la case D
40 SI la case-repère suivante est noire, ALORS ALLER à la séquence 140
50 SI le nombre de la case repérée par la flèche est plus petit que le nombre de la case suivante, ALORS ALLER à la séquence 120
60 EFFACER le nombre de la case D, ECRIRE 1 dans la case D
70 ECRIRE dans la case C le nombre repéré par la flèche
80 EFFACER de la liste B le nombre repéré par la flèche
90 ECRIRE dans la liste B, à la case repérée par la flèche, le nombre situé dans la case suivante de la liste B
100 EFFACER de la liste B le nombre de la case suivante repérée par la flèche
110 ECRIRE dans la liste B, à la case suivante repérée par la flèche, le nombre situé dans la case C

120 EFFACER la flèche de la case-repère, puis DESSINER la flèche dans la case-repère suivante
130 ALLER à la séquence 40
140 SI le nombre de la case D est égale à 0 ALORS ALLER à la séquence 180
150 SI toutes les cases-repère sont noires SAUF la première, ALORS ALLER à la séquence 180
160 NOIRCIR la case-repère où se trouve la flèche
170 ALLER à la séquence 20
180 FIN

L'algorithme qui est décrit pour résoudre ce tri est appelé :

TRI PAR LA METHODE DE LA BULLE

Par analogie à la bulle d'air qui remonte à la surface, le nombre le plus léger remonte en début de liste.

Il est loin d'être performant, et même pour certaines configurations de listes très importantes, dont les éléments sont très désordonnés, s'avère être très long.

Mais il a le mérite d'être simple à comprendre et à programmer. C'est là notre seul souci.

Il existe bien évidemment d'autres méthodes de tri. Elles sont décrites dans d'autres ouvrages.

— L'expérience des deux premiers exercices programmés permet de passer rapidement sur des points plus évidents.

En reprenant l'énoncé du problème, il est question d'une bibliothèque de livres représentée par la liste B.

Nous ne pourrons pas utiliser la structure de l'instruction DATA, car elle ne peut en aucun cas être modifiée au cours de l'exécution d'un programme.

Si vous observez bien les possibilités de l'instruction DIM nous pourrons donner à notre liste B une structure de variable, un peu particulière certes, que l'on nomme **VARIABLE TABLEAU.**

Ce qui nous conduit à écrire comme début de programme

2 DIM B(10)

Nous demandons ainsi à l'ordinateur de nous créer une variable tableau qui aura comme nom B, et qui pourrait s'apparenter aux 10 rayons d'une bibliothèque.

Il faut maintenant stocker des nombres dans cette variable B.

Nous allons demander ce travail à l'ordinateur et lui programmer une séquence d'instructions qui saisira des nombres au clavier. Ensuite, il les rangera dans les cases de la variable tableau B.

Ce qui donne comme instructions

```
4   FL = 0
6   INPUT B (FL)
8   FL = FL + 1
10  IF FL 10 THEN GOTO 6
```

La variable FL servira de flèche de repère.

Ainsi quand le programme s'exécutera, les cases de la variable tableau B contiendront les nombres que vous aurez saisis au clavier.

Nous aurions pu écrire d'une autre façon cette série d'instructions en utilisant une boucle **FOR.**

```
4  FOR FL = 0 TO 9
6  INPUT B (FL)
8  NEXT FL
```

Ce qui est plus élégant et plus économique !

A la séquence 20 de l'exercice 3, nous devons repérer le premier livre de gauche.

Reprenons pour cela la variable **FL** qui nous servira **d'INDICE.**

Initialisons-la à zéro.

```
20  FL = 0
```

A la séquence 30, il nous est demandé de placer zéro dans la variable **D.**

```
30  D = 0
```

Maintenant, nous devons nous poser la question si, en réalité, le livre pointé par la flèche est le dernier. Utilisons par exemple un compteur d'arrêt égal au nombre de livres à trier.

```
15  R = 10
```

Si vous saisissez cette instruction après les instructions 20, 30, *etc. cette dernière sera automatiquement rangée au bon endroit par votre ordinateur, c'est-à-dire entre les lignes* 10 *et* 20

INTERESSANT !!!

Après ces petits préparatifs, nous sommes certains d'avoir une réponse satisfaisante à la question de la séquence 40. Nous l'écrirons en BASIC de cette façon :

```
40  IF (FL + 1) = R THEN GOTO 140
```

La séquence 50 est une comparaison entre le nombre contenu dans la case B, indiqué par la variable FL, et le contenu de la case suivante. Nous pourrions écrire :

50 IF B (FL) < = B (FL + 1) THEN GOTO 120

De la séquence 60 à la séquence 80, une permutation doit être faite. Pour cela nous utiliserons une variable intermédiaire

60 D = 1
70 T = B (FL)
80 B (FL) = B (FL + 1)
90 B (FL + 1) = T

Avançons maintenant d'une case sur la liste B. Il suffira d'ajouter 1 au contenu de la variable FL.

Nous obtiendrons comme instruction :

120 FL = FL + 1

Puis continuons les tests

130 GOTO 40

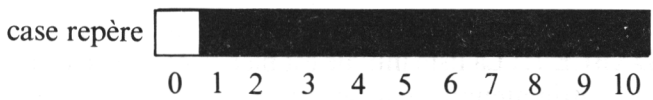
Testons la case D de cette façon :

140 IF D = 0 THEN GOTO 180

La question de la séquence 150 pourrait être formulée de la façon suivante :

150 IF R = 1 THEN GOTO 180

Effectivement. Si la deuxième case du tableau T est noire, nous serons placés dans le cas où toutes les cases sont noires.



Noircir la case repère reviendra à faire — 1 sur la variable R

160 R = R — 1
170 GOTO 20
180 END

Regroupons toutes les instructions pour faire un seul et même programme :

```
2 DIM B(10)
4 FL=0
6 INPUT B(FL)
8 FL=FL+1
10 IF FL < 10 THEN GOTO 6
15 R = 10
20 FL = 0
30 D = 0
40 IF FL+1 = R THEN GOTO140
50 IF B(FL) <= B(FL+1) THEN GOTO 120
55 REM PERMUTATION DE DEUX NOMBRES
60 D=1
70 T=B(FL)
80 B(FL)=B(FL+1)
90 B(FL+1)=T
120 FL=FL+1
130 GOTO 40
135 REM TEST S'IL Y A EU PERMUTATION
140 IF D = 0 THEN GOTO 180
150 R=R-1
160 IF R=1 THEN GOTO 180
170 GOTO 20
180 REM EDITION DES NOMBRES TRIES
200 FOR FL=0 TO 9
210 PRINT B(FL);" "
220 NEXT FL
```

4 FOR FL = 0 TO 9
6 INPUT B (FL)
8 NEXT FL

Programmation plus simple.

Remplace les instructions 4 à 10.

160 IF R < > 1 THEN GOTO 20

170 END

Remplace avantageusement les instructions 160 à 180.

Il est nécessaire de vérifier le bon fonctionnement du programme. Nous pourrions sortir la liste ainsi triée en fin de programme.

180 REM EDITION
200 FOR FL = 0 TO 9
210 PRINT B (FL)
220 NEXT FL

Exécutez ce programme en entrant la commande **RUN**.

FORMIDABLE !!! La liste initiale est bien triée. Vous n'en revenez pas.

Et pourtant l'ordinateur n'a réalisé que ce qui lui a été ordonné.

Cela confirme que l'ordinateur est bien l'esclave de l'homme et qu'il ne peut rien sans lui : comment voulez-vous qu'une machine invente des choses à votre place ?

C'est une question qui aura peut-être une réponse, un jour...

CHAPITRE **6**

Du fonctionnel à la programmation

Changeons maintenant de procédé et ayons une attitude plus classique devant les problèmes que vous rencontrerez pour programmer.

Ensemble, nous allons traiter une série d'exercices. Nous les analyserons puis nous les coderons en BASIC.

6.1 LES IMPÔTS

Quand vous recevez l'enveloppe annuelle et fatidique des contributions directes pour déclarer vos revenus, il vous est dit que les calculs ne sont pas à faire car l'ordinateur central du service des impôts le fera à votre place.

Malgré tout, vous aimeriez être renseigné sur les sommes que vous allez payer l'année prochaine. Cela vous permettra de vérifier ce que l'on vous demande ou tout simplement de budgéter vos dépenses.

Pour cela l'imprimé n° 2041 vous est adressé et vous permettra, en suivant scrupuleusement le travail à faire, d'aboutir au résultat tant attendu... Nous vous proposons de reprendre cet imprimé pour réaliser un petit programme qui vous apportera aide et assistance dans l'accomplissement de cette tâche pénible.

Imprimé n° 2041

2 - Déterminez votre nombre de parts (N)

vous avez / vous êtes	aucune pers. à charge		nombre d'enfants à charge (1) (2)										
	cas général	cas particuliers (3)	1	2	3	4	5	6	7	8	9	10	et ainsi de suite en ajoutant une demi part
Marié (4)	2	»	2,5	3	4	4,5	5	5,5	6	6,5	7	7,5	
Célibataire (6) Div. (6) Veuf (5)(6)	1	1,5	2	2,5	3,5	4	4,5	5	5,5	6	6,5	7	

(1) Si vous avez à votre charge une personne invalide, autre qu'un enfant, ajoutez 1 part par personne au nombre de parts ci-dessus.
(2) Le nombre de parts est augmenté de 1/2 part par enfant à charge titulaire de la carte d'invalidité.
(3) Vous avez coché une, ou même plusieurs des cases P, E, K, S, W du cadre II de la page 2 de la déclaration (la 1/2 part supplémentaire ne peut être accordée qu'une seule fois).
(4) Ajoutez 1/2 part lorsque vous ou votre conjoint êtes invalide (1 part si chacun est invalide).
(5) Si vous avez coché la case L du cadre II de la page 2 de la déclaration, ajoutez 1/2 part.
(6) Si vous avez coché la case P du cadre II de la page 2 de la déclaration, ajoutez 1/2 part lorsque vous avez des charges de famille.

Vous n'avez pas d'impôt à acquitter si :

— votre revenu net de frais professionnels n'excède pas 27 000 F (ou 29 500 F si vous avez plus de 65 ans).
— votre revenu global (ligne 13 de la fiche de calculs), augmenté de l'abattement de 20 % *ligne h*, n'excède pas 22 415 F.
— votre revenu imposable (R) arrondi à la dizaine inférieure n'excède pas les limites du tableau ci-dessous.

Pour	Votre revenu (R) n'excède pas :	Pour	Votre revenu (R) n'excède pas :	Pour	Votre revenu (R) n'excède pas :	Pour	Votre revenu (R) n'excède pas :	Pour	Votre revenu (R) n'excède pas :
1 part	25 210	2 parts	28 500	3 parts	41 400	4 parts	54 310	5 parts	67 210
1,5 part	25 280	2,5 parts	34 950	3,5 parts	47 860	4,5 parts	60 760	5,5 parts	73 670

3 - Calculez le quotient " QF " $\left(\text{quotient familial} = \frac{\text{R (revenu imposable)}}{\text{N (nombre de parts)}}\right)$

4 - Calculez votre impôt (I)

Appliquez le barème ci-contre

Si QF est inférieur ou égal à	12 620 F		votre impôt sera égal à : 0
Si QF est supérieur à	12 620 F	et inférieur ou égal à 13 190 F	votre impôt sera égal à : (R × 0,05) – (631 F × N)
Si QF est supérieur à	13 190 F	et inférieur ou égal à 15 640 F	votre impôt sera égal à : (R × 0,10) – (1 290,5 F × N)
Si QF est supérieur à	15 640 F	et inférieur ou égal à 24 740 F	votre impôt sera égal à : (R × 0,15) – (2 072,5 F × N)
Si QF est supérieur à	24 740 F	et inférieur ou égal à 31 810 F	votre impôt sera égal à : (R × 0,20) – (3 309,5 F × N)
Si QF est supérieur à	31 810 F	et inférieur ou égal à 39 970 F	votre impôt sera égal à : (R × 0,25) – (4 900 F × N)
Si QF est supérieur à	39 970 F	et inférieur ou égal à 48 360 F	votre impôt sera égal à : (R × 0,30) – (6 898,5 F × N)
Si QF est supérieur à	48 360 F	et inférieur ou égal à 55 790 F	votre impôt sera égal à : (R × 0,35) – (9 316,5 F × N)
Si QF est supérieur à	55 790 F	et inférieur ou égal à 92 970 F	votre impôt sera égal à : (R × 0,40) – (12 106 F × N)
Si QF est supérieur à	92 970 F	et inférieur ou égal à 127 860 F	votre impôt sera égal à : (R × 0,45) – (16 754,5 F × N)
Si QF est supérieur à	127 860 F	et inférieur ou égal à 151 250 F	votre impôt sera égal à : (R × 0,50) – (23 147,5 F × N)
Si QF est supérieur à	151 250 F	et inférieur ou égal à 172 040 F	votre impôt sera égal à : (R × 0,55) – (30 710 F × N)
Si QF est supérieur à	172 040 F	et inférieur ou égal à 195 000 F	votre impôt sera égal à : (R × 0,60) – (39 312 F × N)
Si QF est supérieur à	195 000 F		votre impôt sera égal à : (R × 0,65) – (49 062 F × N)

ANALYSE FONCTIONNELLE

C'est un problème de moins à résoudre dont la solution nous est donnée par l'administration fiscale. Il suffira de suivre strictement les indications 2, 3 et 4 de la page 7.

ANALYSE ORGANIQUE

Il faut l'écrire, elle n'existe pas. Reprenons l'imprimé page 7.

Si l'on observe l'ensemble, il paraît bien confus. Des tableaux, des formules, des conditions particulières, puis d'autres encore.

Comment faire pour arriver à quelque chose qui soit programmable ?

Pas de panique. Nous vous proposons, étape par étape, d'analyser la situation et de dégager les grandes lignes du travail à faire.

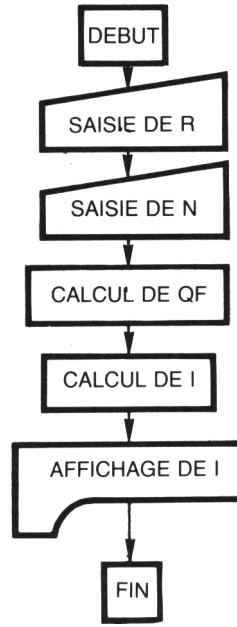
Nous supposerons que le calcul du **MONTANT DU REVENU NET IMPOSABLE** est déjà fait. Plus tard, nous verrons s'il n'y a pas moyen de le calculer automatiquement. De toute façon cette valeur doit être saisie sur votre ordinateur. Nous la nommerons R.

Il faudra aussi déterminer le nombre de parts. C'est une autre valeur à saisir. Nous la nommerons **N.**

Les paramètres personnels sont maintenant disponibles. Il ne nous reste plus qu'à appliquer l'algorithme de calcul du **QUOTIENT FAMILIAL QF** et le **MONTANT DE L'IMPOT I.**

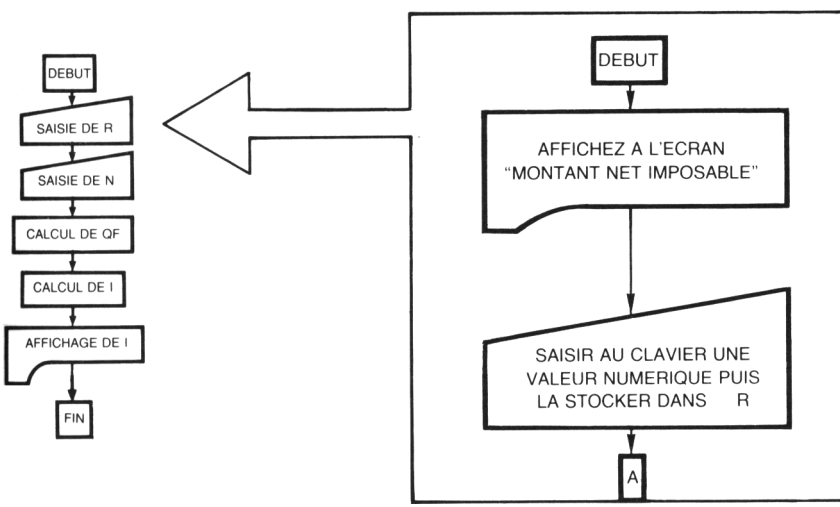
Ces calculs terminés, nous procéderons à l'affichage du résultat.

Nous pouvons dès à présent écrire l'organigramme général :

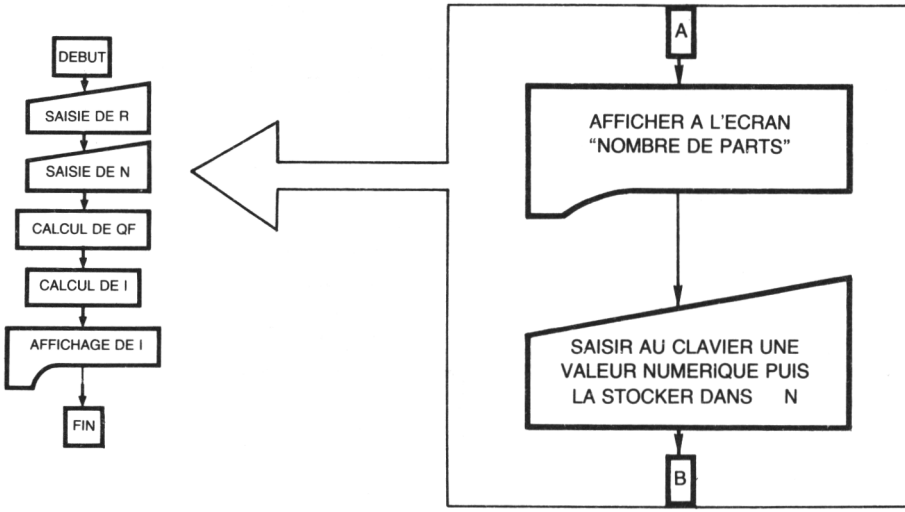


Reprenons chaque pavé de l'organigramme afin de détailler les fonctions.

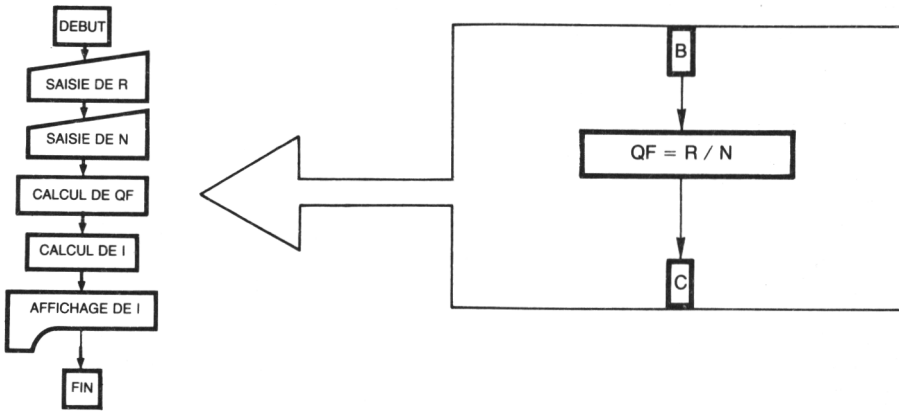
SAISIE DE R : C'est aussi prévoir l'affichage d'un message qui renseignera l'utilisateur du programme. Ensuite, il faudra stocker cette valeur afin de la traiter dans les phases suivantes.



SAISIE DE N : Même raisonnement que pour la saisie du montant.



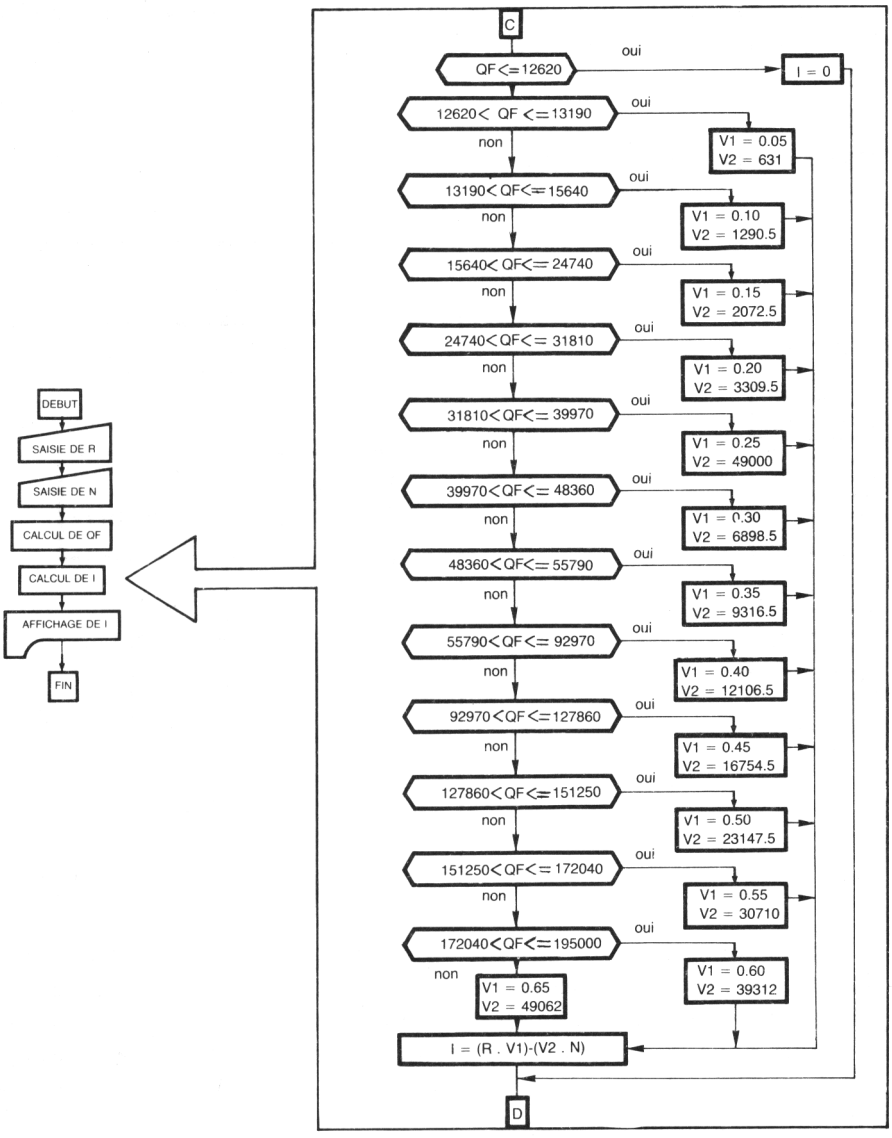
CALCUL DE QF : C'est l'application de la formule proposée au chapitre 3.



CALCUL DE I : Le calcul est un peu plus complexe à déterminer, et va demander une petite préparation avant d'appliquer les formules proposées au chapitre 4.

Nous constatons, d'après l'analyse fonctionnelle, que la valeur I dépendra de deux paramètres.

Le premier paramètre **QF** : Il est déjà calculé à l'étape précédente.

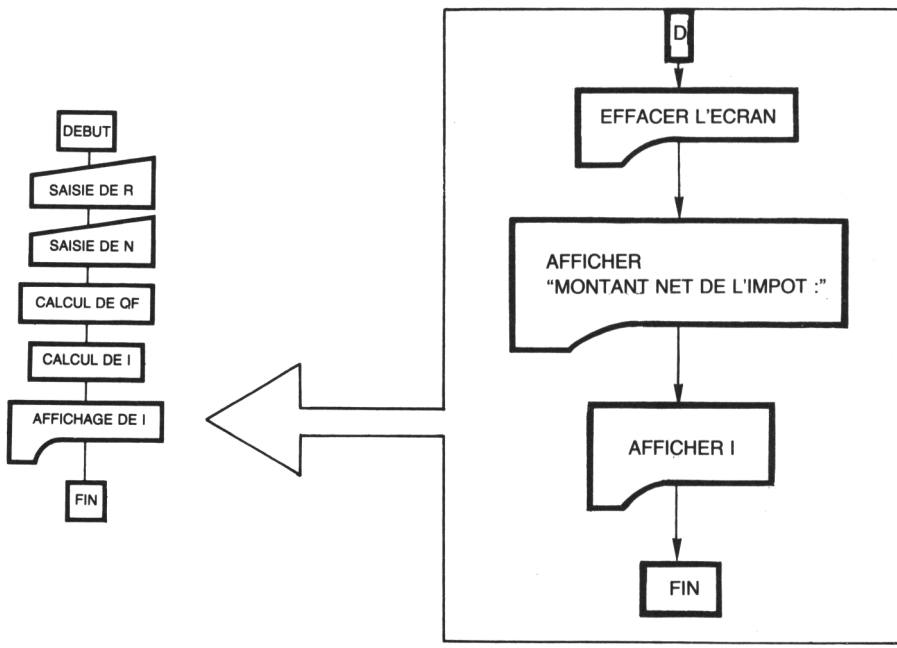


Mais on s'aperçoit qu'en fonction des valeurs que QF prendra, l'allure générale de la formule ne change pas. Elle sera toujours de la forme :

$$\boxed{I = (R.v1) - (v2.N)}$$

Il suffira d'effectuer une série de tests pour calculer les valeurs de v1 et v2, puis d'appliquer la formule ci-dessus.

ÉDITION DU RÉSULTAT :



L'analyse organique est terminée. Passons au codage du programme.

Pour la commodité de lecture du programme, nous allons ajouter des lignes de commentaire. Cette façon de programmer vous apportera plus de confort pour la détection des éventuelles erreurs. Vous aurez moins de mal à vous replonger dans la programmation, si vous devez faire des modifications plus tard.

```
10 REM CALCUL DE L'IMPOT
20 REM
30 REM SAISIE DU MONTANT NET
40 CLS
50 PRINT"MONTANT NET IMPOSABLE :";
60 INPUT R
65 REM
70 REM SAISIE DU NBRE DE PART
80 PRINT"NOMBRE DE PARTS:";
90 INPUT N
95 REM
100 REM CALCUL DU QUOTIENT FAMILIAL
105 REM
110 QF=R/N
115 REM
120 REM CALCUL DES VALEURS V1 ET V2
125 REM
130 IF QF > 12620 THEN GOTO 160
140 I=0
150 GOTO 1010
160 IF QF > 13190 THEN GOTO 190
170 V1=0.05 : V2=631
180 GOTO 1000
190 IF QF > 15640 THEN GOTO 220
200 V1=0.1  : V2=1290.5
210 GOTO 1000
220 IF QF > 24740 THEN GOTO 250
230 V1=0.15 : V2=2072.5
240 GOTO 1000
250 IF QF > 31810 THEN GOTO 280
260 V1=0.2  : V2=3309.5
270 GOO 1000
280 IF QF > 39970 THEN GOTO 310
290 V1=0.25 : V2=4900
300 GOTO 1000
310 IF QF > 48360 THEN GOTO 340
320 V1=0.3  : V2=6898.5
330 GOTO 1000
340 IF QF > 55790 THEN OTO 370
350 V1=0.35 : V2=9316.5
360 GOTO 1000
370 IF QF > 92970 THEN GOTO 400
380 V1=0.4  : V2=12106
390 GOTO 1000
400 IF QF > 127860 THEN GOTO 430
410 V1=0.45 : V2=16754.5
420 GOTO 1000
430 IF QF > 151250 THEN GOTO 460
440 V1=0.5  : V2=23147.5
450 GOTO1000
460 IF QF > 172040 THEN GOTO 490
470 V1=0.55 : V2=30710
480 GOTO1000
```

```
490 IF QF > 195000 THEN GOTO 520
500 V1=0.6  : V2=39312
510 GOTO1000
520 V1=0.65 : V2=49062
1000 I=(R*V1)-(V2*N)
1010 REM
1020 REM EDITION DU RESULTAT
1030 REM
1040 PRINT"MONTANT NET DE L'IMPOT = ";I
1050 END
```

6.2 LE LOTO

Que pourrions-nous écrire comme programme BASIC pour inventer des nombres aléatoires de type LOTO NATIONAL ?

ANALYSE FONCTIONNELLE

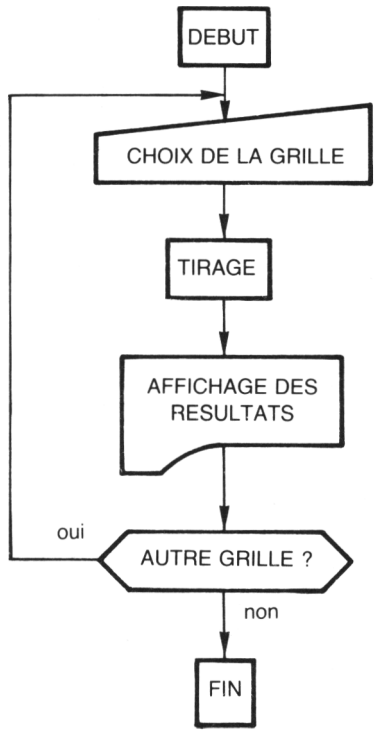
Les nombres tirés aléatoirement devront être compris entre 1 et 49. Ils devront être tous différents pour une même grille.

Ensuite il faudra éditer des séries de 6 à 10 nombres, les grilles du loto pouvant être jouées ainsi. Cela implique donc un choix de grille avant le tirage des n NOMBRES.

Les principaux éléments sont connus, il ne nous reste plus qu'à ébaucher une analyse organique.

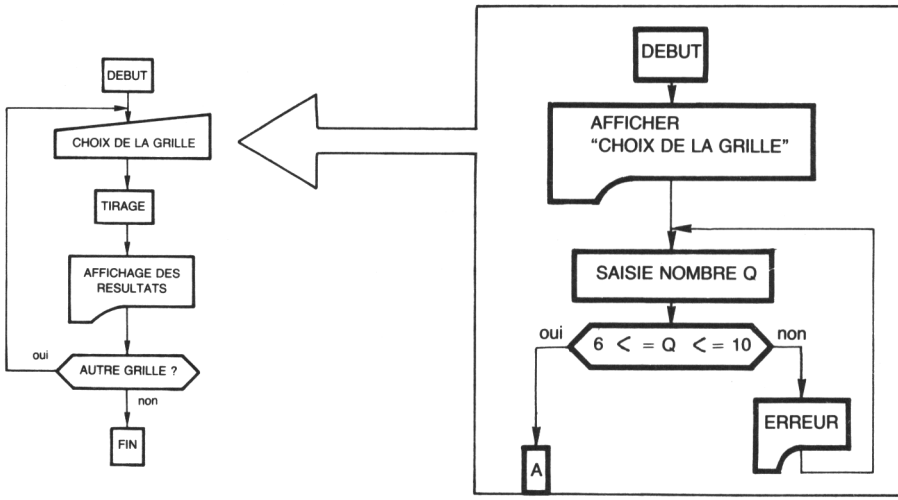
ANALYSE ORGANIQUE

Commençons par dessiner l'organigramme général.



Reprenons chaque pavé de l'organigramme, et définissons une analyse plus fine.

CHOIX DE LA GRILLE



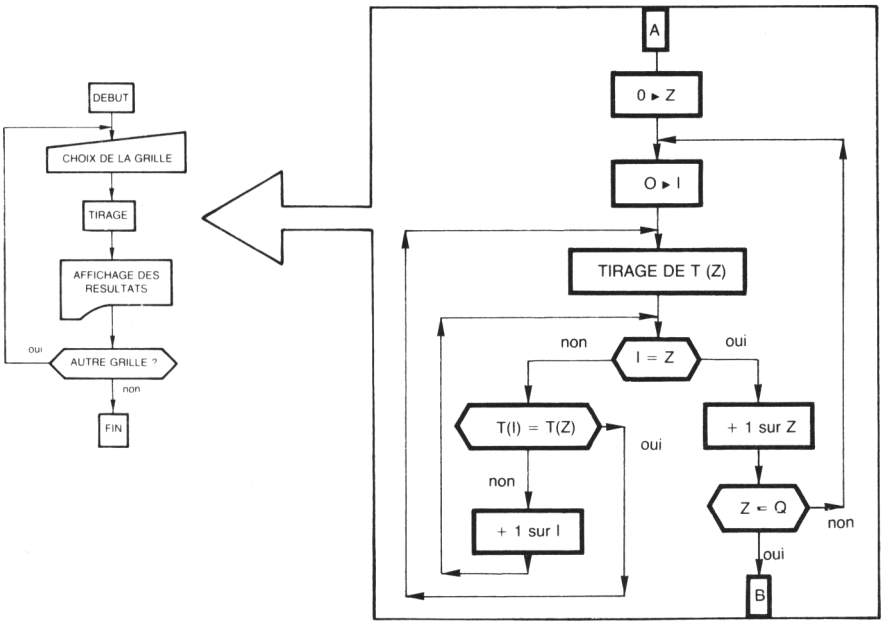
TIRAGE ALÉATOIRE

La valeur Q, saisie et contrôlée à l'étape précédente, permettra de répéter Q fois le tirage d'un nombre aléatoire. Chaque nombre ainsi obtenu sera comparé avec les précédents nombres de la série en cours, de façon à éliminer les doubles éventuels. Si double il y a, alors nous procéderons à un autre tirage, et ainsi de suite, jusqu'à obtention du résultat escompté. Si le nombre tiré n'a pas de double, alors il sera stocké dans la série. Nous obtenons comme détail l'organigramme ci-contre.

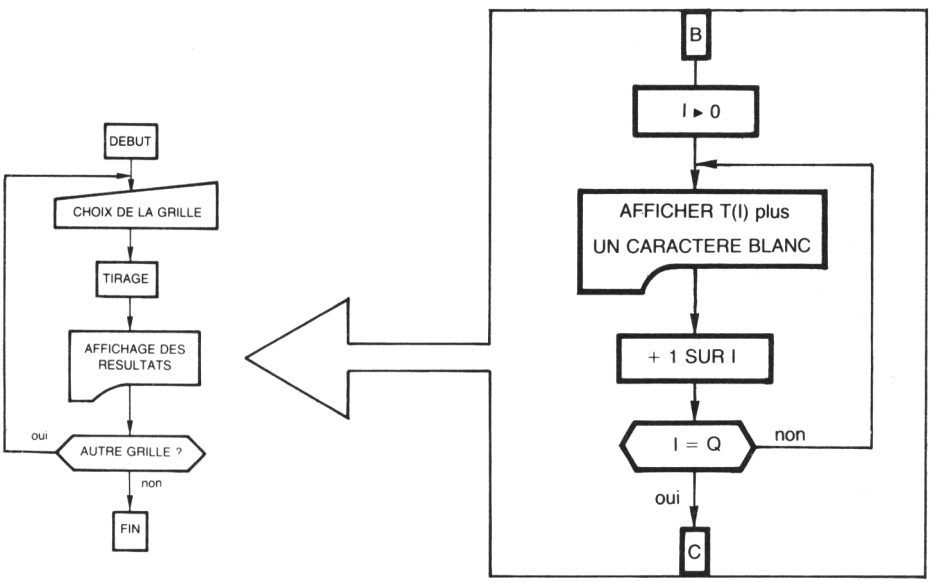
Donnons quelques explications sur les moyens mis en œuvre pour obtenir un nombre aléatoire compris entre 1 et 49.

Ce qu'il est nécessaire de savoir : En BASIC, l'instruction **Z = RND(X)** permet de retourner dans Z un nombre réel compris entre 0 et 1. Jusque-là, pas de problème. Mais c'est insuffisant. Effectivement, il nous faut trouver le moyen d'obtenir des nombres entiers, aléatoires et compris dans des limites plus grandes.

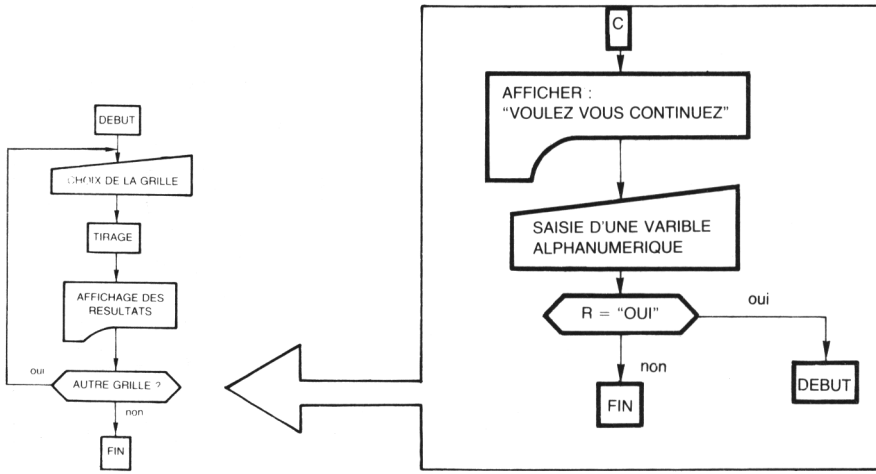
Si nous multiplions par 49 ce premier résultat, nous obtiendrons un nombre aléatoire compris entre 0 et 48.99999. . . . Il suffira donc d'ajouter 1 puis de ne conserver que la partie entière.



AFFICHAGE DE LA GRILLE OBTENUE



PRÉPARONS LA SÉQUENCE DE FIN



PROGRAMMATION

```
5 CLS
10 REM TIRAGE DU LOTO
15 REM
20 DIM T(9)
30 PRINT "CHOIX DE LA GRILLE :";
40 INPUT Q
50 IF Q>=6 AND Q<=10 THEN GOTO 100
60 PRINT"ERREUR"
70 GOTO 30
75 REM
80 REM
85 REM TIRAGE DES Q NOMBRES
90 REM
100 Z=0
110 I=0
120 X=INT(RND(1)*49)+1
130 IF I=Z THEN GOTO 170
140 IF T(I)=X THEN GOTO 110
150 I=I+1
160 GOTO 130
170 T(I)=X
180 Z=Z+1
```

```
190 IF Z < Q THEN GOTO 110
200 REM
210 REM AFFICHAGE DES Q NOMBRES TIRES
220 RE
230 FOR I=0 TO Q-1
240 PRINT T(I);" ";
250 NEXT I
260 REM
270 REM DEMANDE D'UNE NOUVELLE GRILLE
280 REM
290 PRINT:PRINT"AUTRE GRILLE  ( OUI / NON )"
300 INPUT R$
310 IF R$="OUI" THEN GOTO 30
320 END
```

6.3 PRÉFIXES ET RADICAUX

HYPERGENIAL, SUPERMECHANT sont des associations de « mots » ou :

HYPER et SUPER sont des « PREFIXES »
GENIAL et MECHANT sont des « RADICAUX »

Nous pourrions à loisir inventer des mots formés de cette façon comme :

SUPERGENIAL ou HYPERMECHANT etc...

Nous vous proposons, dans cet exercice, de confier à l'ordinateur le soin de faire ces inventions au gré du hasard. Les résultats sont souvent étonnants, voire comiques. Voici deux listes de préfixes et de radicaux. Vous pouvez les compléter à loisir

PREFIXES : Hyper, super, méga, maxi, mini, rétro, extra, etc...
RADICAUX : Génial, méchant, gentil, souple, vieux, jeune, etc...

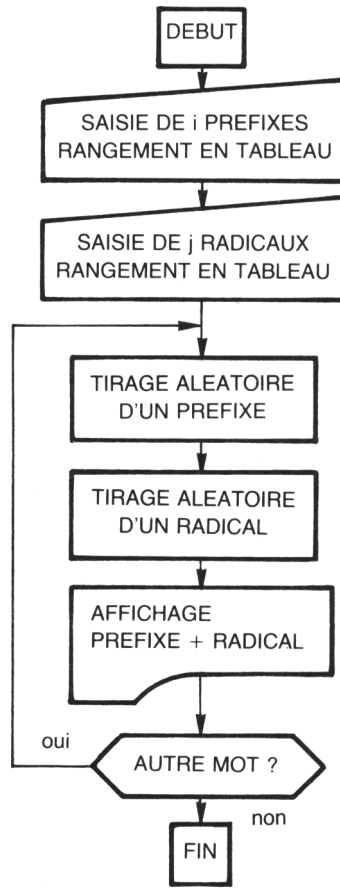
ANALYSE FONCTIONNELLE

Nous vous proposons d'écrire un programme qui associera un préfixe tiré parmi i, à un radical tiré parmi j, le tirage étant aléatoire.

Les i préfixes et les j radicaux seront saisis au clavier, puis stockés dans des tableaux de variables du type chaîne de caractères.

ANALYSE ORGANIQUE

Essayons d'établir un organigramme général de l'analyse fonctionnelle proposée.

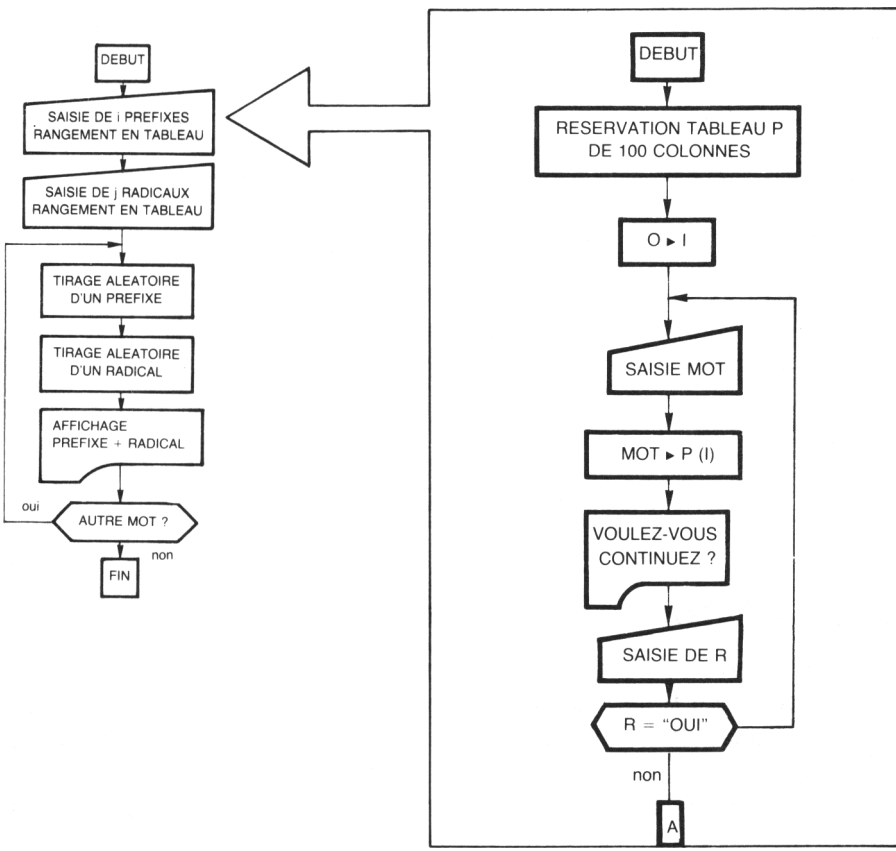


DETAIL DE L'ANALYSE ORGANIQUE

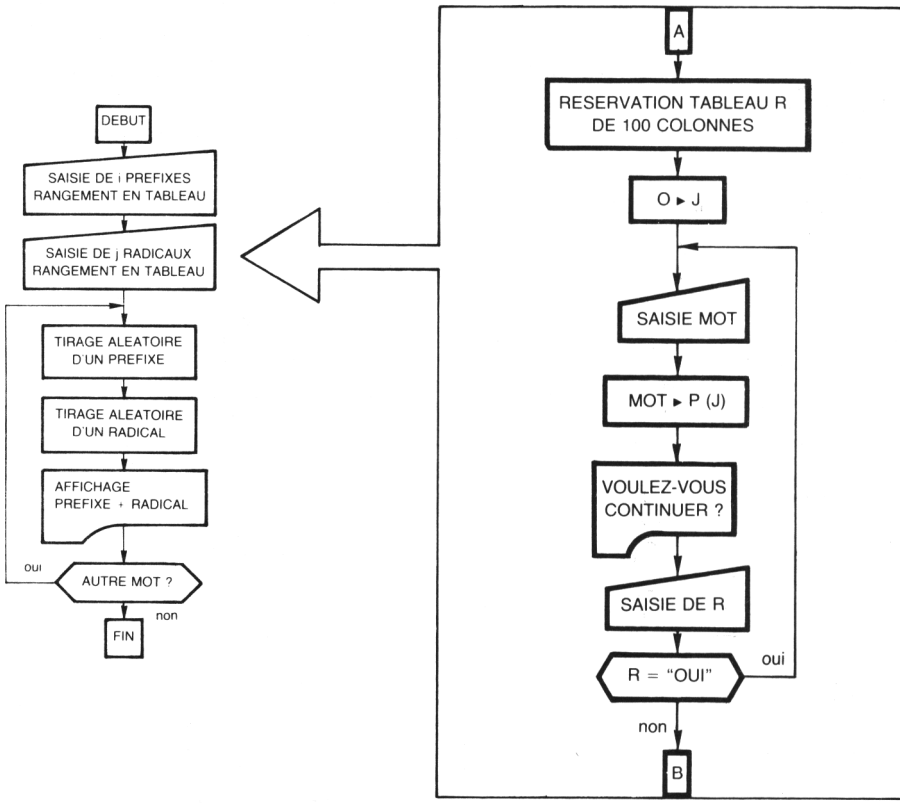
Il va falloir saisir une série de mots que nous rangerons dans une variable de type « tableau à une dimension » (1 ligne, n colonnes). A chaque mot saisi, nous devrons poser une question de continuité ou d'arrêt du chargement du tableau.

La variable tableau se nommera P et aura une dimension arbitraire de 100 colonnes.

Nous obtenons comme organigramme détaillé :



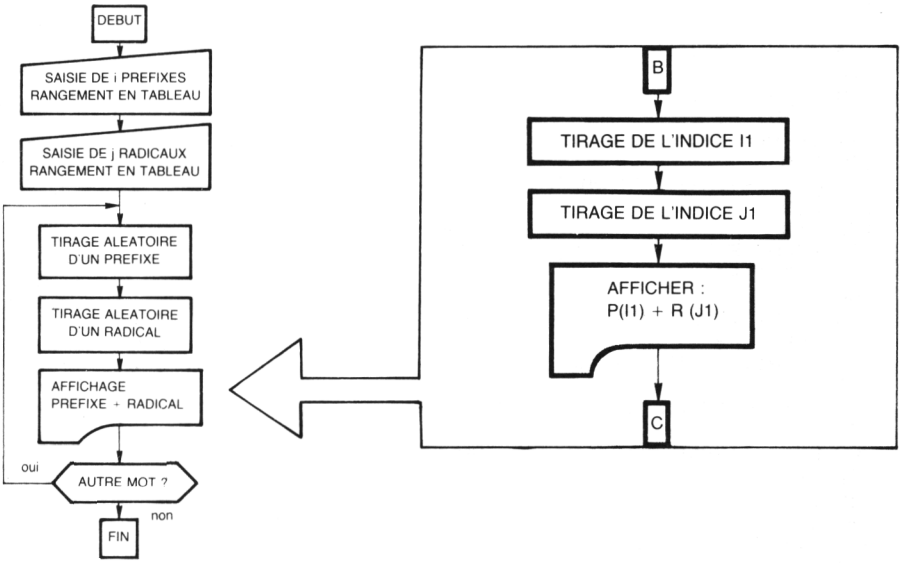
Pour la saisie des radicaux nous allons procéder exactement de la même façon :



TIRAGE ALÉATOIRE D'UN PRÉFIXE

D'après l'exercice précédent, le problème est en partie résolu. Il faut cette fois-ci procéder au tirage aléatoire d'un nombre compris entre 0 et I – 1. Ce nombre nous servira d'indice dans le tableau P pour extraire un préfixe au hasard.

Le procédé sera le même pour le tirage des radicaux, la limite des nombres tirés sera comprise entre 0 et J – 1.



AUTRE MOT

Nous devons traiter la condition de continuité ou d'arrêt d'édition des mots obtenus. Cette phase d'analyse ne doit plus vous poser de problème, puisqu'elle a fait l'objet d'analyses traitées lors des exercices précédents. Nous coderons directement cette partie.

PROGRAMMATION

```
5 REM PREFIXES ET RADICAUX
10 CLS
20 REM
25 REM DECLARATION DES 2 TABLEAUX
30 REM
40 DIM P$(100)
50 DIM R$(100)
60 REM
65 REM  SAISIE DES PREFIXES
70 REM
80 I=0
90 PRINT"ENTREZ UN PRFIXE";
```

```
100 INPUT P$(I)
110 I=I+1
120 PRINT"VOULEZ VOUS CONTINUER";
130 INPUT A$
140 IF A$="OUI" THEN GOTO 90
150 REM
155 REM  SAISIE DES RADICAUX
160 REM
170 J=0
180 PRINT"ENTREZ UN RADICAL";
190 INPUT R$(J)
200 J=J+1
210 PRINT"VOULEZ-VOUS CONTINUER";
220 INPUT A$
230 IF A$="OUI" THEN GOTO 180
240 REM
245 REM  EDITION DES MOTS INVENTES
250 REM
260 REM  TIRAGE DES INDICES TABLEAU
270 REM
280 CLS
290 I1=INT(RND(1)*I)
300 J1=INT(RND(1)*J)
310 PRINTP$(I1)+R$(J1)
320 REM
325 REM QUESTION  ARRET OU FN
330 REM
340 PRINT"AUTRE MOT  ( OUI/NON )
350 INPUT R$
360 IF R$="OUI" THEN GOTO 290
380 END
```

Pour simplifier la partie d'édition des mots inventés par le tirage aléatoire, nous vous proposons de ne plus poser la question « **AUTRE MOT** » après l'affichage, et de convenir que — tant que l'utilisateur enfoncera la touche **BARRE D'ESPACE** — le programme repartira sur la séquence d'affichage d'un nouveau mot. Quand une touche différente sera enfoncée, alors le programme s'arrêtera.

Nous obtiendrons comme nouvelle programmation :

340 GET A$
350 IF A$ = " " THEN GOTO 190
380 END

Ces trois instructions remplacent les instructions 230 à 270.

```
5 REM PREFIXES ET RADICAUX
10 CLS
20 REM
25 REM DECLARATION DES 2 TABLEAUX
30 REM
40 DIM P$(100)
50 DIM R$(100)
60 REM
65 REM  SAISIE DES PREFIXES
70 REM
80 I=0
90 PRINT"RENTREE EN TABLEAU DES PREFIXES"
100 INPUT P$(I)
110 IF P$(I)="FIN" THEN GOTO 150
120 I=I+1
130 GOTO 100
150 REM
155 REM  SAISIE DES RADICAUX
160 REM
170 J=0
180 PRINT"RENTREE EN TABLEAU DES RADICAUX"
190 INPUT R$(J)
200 IF R$(J)="FIN" THEN GOTO 240
210 J=J+1
220 GOTO 190
240 REM
245 REM  EDITION DES MOTS INVENTES
250 REM
260 REM  TIRAGE DES INDICES TABLEAU
270 REM
280 CLS
290 I1=INT(RND(1)*I)
300 J1=INT(RND(1)*J)
310 PRINTP$(I1)+R$(J1)
320 REM
325 REM CONTINUITE DU PROG TANT QUE
330 REM BARRE ESPACE ENFONCEE
340 GET A$
350 IF A$=" " THEN GOTO 290
380 END
```

6.4 LES NOMBRES PREMIERS

Nous vous proposons d'apprendre à définir, à partir d'une définition mathématique, un **ALGORITHME** puis un programme BASIC pour éditer des nombres premiers.

FONCTIONNEL

Afficher à l'écran les cent premiers nombres premiers positifs obtenus par l'application de la définition ci-dessous.

Définition : un nombre est dit « premier » quand il n'est divisible par aucun nombre premier précédent sauf par 1 et par lui-même.

Pour la facilité d'écriture, nous n'appliquerons pas strictement cette règle car elle présente l'inconvénient de stocker les nombres premiers précédemment obtenus, sachant que vous auriez très rapidement des problèmes d'encombrement mémoire. Nous nous assurerons qu'un nombre N ne soit divisible par aucun des entiers positifs compris entre 2 et N – 1. Auquel cas le nombre N sera premier. En procédant ainsi, nous n'aurons aucun problème de mémoire, ce qui nous permettra de calculer si des nombres importants sont premiers. Par contre, nous perdrons en rapidité.

ANALYSE ORGANIQUE

Commençons par définir une ébauche d'algorithme.

Nous allons incrémenter (= ADDITIONNER), par pas de 1, le contenu d'une variable numérique appelée N.

Nous testerons si N contient un nombre premier :

Si non, nous passerons au nombre suivant.

Si oui, nous éditerons le contenu de N.

A chaque édition de N, nous comptabiliserons +1 dans un compteur d'édition nommé V.

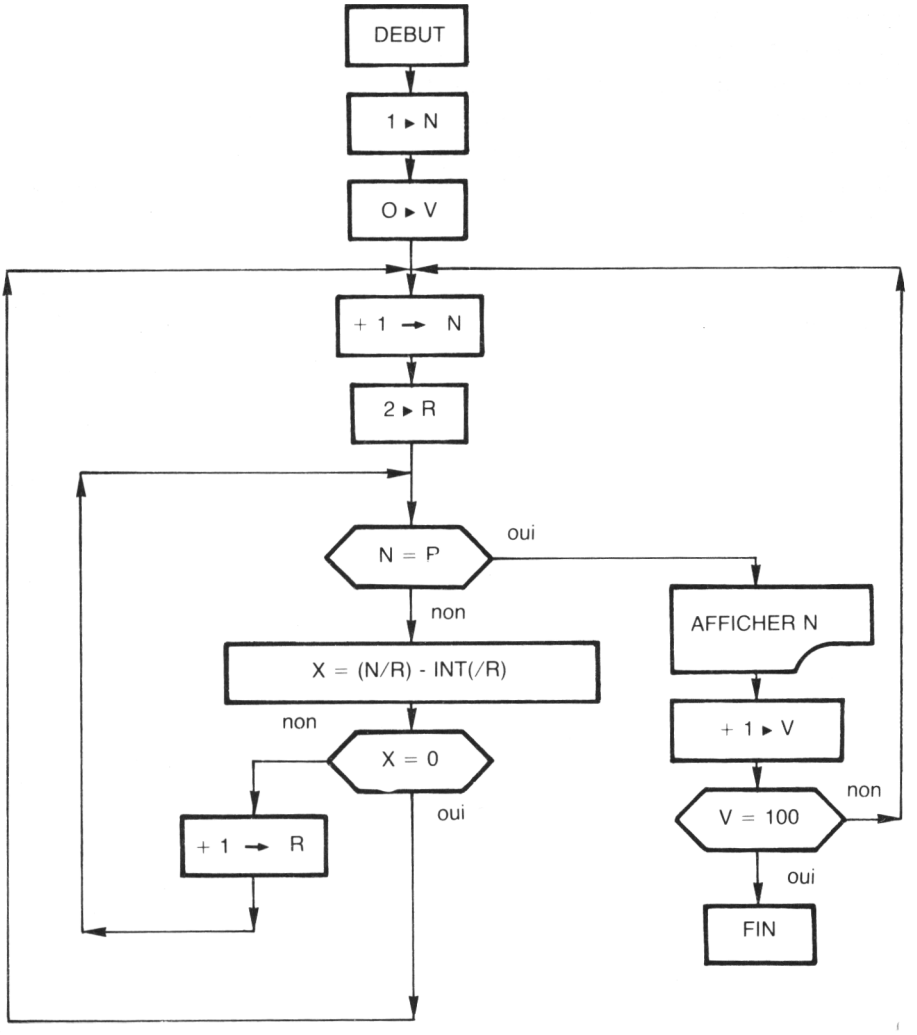
Si V est égal à la limite fixée, alors nous arrêterons le programme.

Si V est inférieur, alors nous passerons au nombre suivant.

Puisque N commence à 1 *et sachant que* 1 *est premier, nous commencerons par l'édition de N puis nous rentrerons dans le processus de détermination cité ci-dessus.*

Dessin de l'organigramme

N	est la variable contenant le nombre à tester.
V	est la variable compteur d'édition.
R	est la variable progressant de 2 à N – 1.
X	est une variable intermédiaire de calcul.



Quelques explications sur la formule $\boxed{\mathbf{X = (N/R) - INT(N/R)}}$.

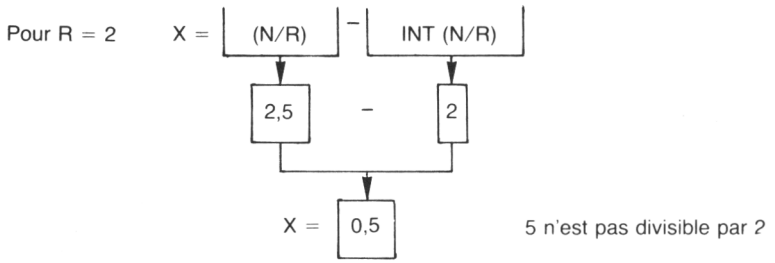
Elle permet de vérifier si la division de N par R donne un reste non nul.
Si c'est le cas, alors le nombre N n'est pas divisible par R.
Si le reste est égal à zéro, alors N est divisible par R.
Nous vous rappelons qu'en BASIC les variables comme N, R etc... qui

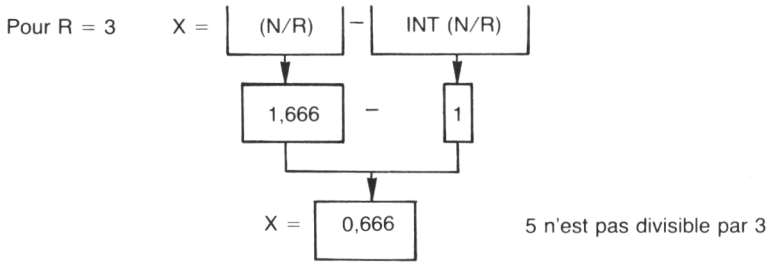
ne sont pas suivies d'un $ sont des variables de type REEL. C'est-à-dire qu'elles ont une partie entière et une partie fractionnaire.

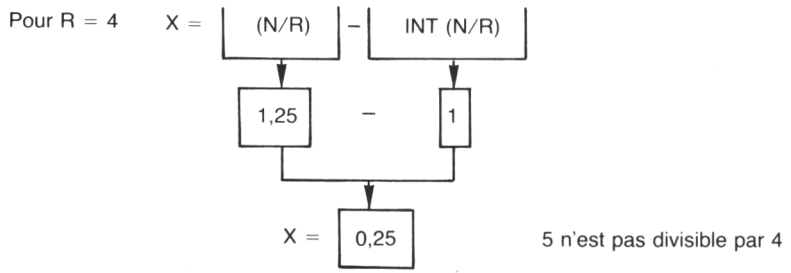
En analysant une partie de la formule (N/R) nous obtiendrons après exécution de la division un résultat de type REEL (il y aura une partie fractionnaire). Pour ne conserver que la partie fractionnaire du résultat obtenu, il est nécessaire d'en soustraire la partie entière. C'est chose faite avec INT(N/R).

Exemple :

Soit N = 5







Le nombre 5 est donc un nombre premier

ÉCRITURE DU PROGRAMME

```
5 CLS
10 REM NOMBRES PREMIERS
15 REM CAS OU 1 N'EST PAS PREMIER
20 REM
25 REM 2 ECOLES CHEZ LES MATHEMATICIENS
30 REM INITIALISATION DES VARIABLES
40 N=1
50 V=0
60 N=N+1
70 R=2
80 IF N=R THEN GOTO 130
90 X=(N/R)-INT(N/R)
100 IF X=0THEN GOTO 60
110 R=R+1
120 GOTO 80
130 PRINTN;" ";
140 V=V+1
150 IF V <> 100 THEN GOTO 60
160 END
```

6.5 JEU DE CARTES

Nous vous proposons de réfléchir à la distribution des cartes au cours d'une partie de bridge.

Pour des raisons évidentes, nous n'irons pas jusqu'à analyser le jeu même de la carte et encore moins celui des annonces. Les règles sont trop complexes.

ANALYSE FONCTIONNELLE

Que connaissons-nous du problème posé ?

— Le jeu de bridge est constitué de 52 cartes réparties de la façon suivante :

13 trèfles, 13 carreaux, 13 cœurs, 13 piques.

— 4 joueurs reçoivent à tour de rôle 13 cartes.

— Le donneur possède, avant la distribution, un paquet de 52 cartes.

Il prend soin de bien le mélanger afin que chaque joueur ait la même chance que les trois autres de recevoir de bonnes cartes.

— La distribution se fait carte par carte, donnée à tour de rôle aux quatre joueurs.

— Enfin, la partie peut commencer, mais là s'arrête notre étude.

ANALYSE ORGANIQUE

Que devons-nous préparer et quels outils allons-nous utiliser pour faire réaliser cette distribution par un programme ?

— Le problème de la codification des 52 cartes :

Nous pourrions associer une hauteur de carte à un numéro ou à une lettre, et la couleur de cette carte, à un caractère reprogrammé de façon à obtenir le dessin de la couleur.

Ensuite nous accolerons ce numéro ou cette lettre au caractère reprogrammé, en insérant un blanc au milieu pour la facilité de lecture.

Table de correspondance hauteur de la carte et numéro — ou lettre — associée

AS	= 1	SEPT	= 7
ROI	= R	SIX	= 6
REINE	= Q	CINQ	= 5
VALET	= V	QUATRE	= 4
DIX	= 10	TROIS	= 3
NEUF	= 9	DEUX	= 2
HUIT	= 8		

Reprogrammation des caractères pour les quatre couleurs.

Nous choisissons arbitrairement le caractère A que nous allons reprogrammer.

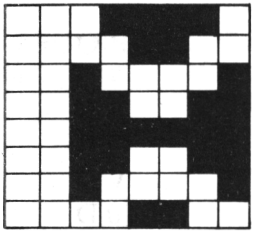
Attention, les deux colonnes de gauche sont ignorées.

Nous verrons plus loin comment modifier en mémoire et aux bons endroits les valeurs calculées pour les quatre caractères.

Note de l'auteur : Nous avons particulièrement apprécié la facilité de reprogrammation des caractères sur l'ORIC et sur le TO7.

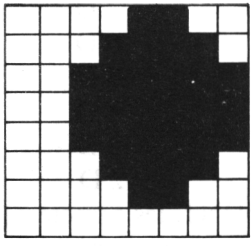
IMAGES POSSIBLES

La couleur TREFLE :
CARACTERE A



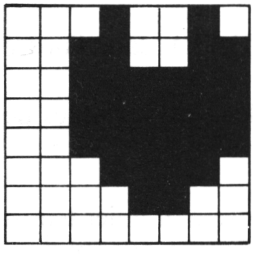
VALEURS BINAIRES	VALEURS DECIMALES
011110	30
001100	12
100001	33
110011	51
111111	63
110011	51
100001	33
001100	12

La couleur CARREAU :
CARACTERE B



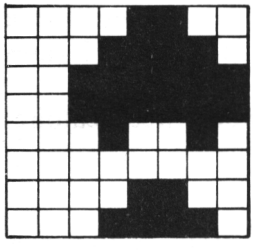
001100	12
011110	30
111111	63
111111	63
111111	63
011110	30
001100	12
000000	0

La couleur CŒUR :
CARACTERE C



010010	18
110011	51
111111	63
111111	63
111111	63
011110	30
001100	12
000000	0

La couleur PIQUE :
CARACTERE D



001100	12
011110	30
111111	63
111111	63
010010	18
000000	0
001100	12
011110	30

D'après les deux tables **HAUTEUR ET COULEUR,** nous mémoriserons le jeu des 52 cartes dans la zone des données fixes nommée DATA. Nous aurons ainsi à ranger en DATA les valeurs suivantes :

LES TREFLES	LES CARREAUX	LES COEURS	LES PIQUES
1 A	1 B	1 C	1 D
R A	R B	R C	R D
Q A	Q B	Q C	Q D
V A	V B	V C	V D
10 A	10 B	-10 C	10 D
9 A	9 B	9 C	9 D
8 A	8 B	8 C	8 D
7 A	7 B	7 C	7 D
6 A	6 B	6 C	6 D
5 A	5 B	5 C	5 D
4 A	4 B	4 C	4 D
3 A	3 B	3 C	3 D
2 A	2 B	2 D	2 D

Sachant qu'à l'édition des cartes par l'instruction PRINT, les caractères reprogrammés apparaîtront sous la forme définie suivant nos spécifications.

— **La main du donneur**

Elle recevra les cartes codifiées issues de la zone DATA dans l'ordre où elles y ont été écrites.

Nous allons considérer cette main comme un tableau de variables de type « chaîne de caractères ».

Pourquoi tableau de variable ?

Parce que le nombre de cartes codifiées va varier dans un premier temps de 0 à 51 au cours du remplissage puis de 51 à 0 pendant la distribution.

— **Les mains de chaque joueur**

Ce seront 4 tableaux de variables qui auront la même structure que le tableau « main du donneur », à la différence près qu'il y aura 13 cases par main au lieu de 52 cases.

Chaque tableau recevra à tour de rôle une valeur codée, tirée aléatoirement du tableau représentant la main du donneur.

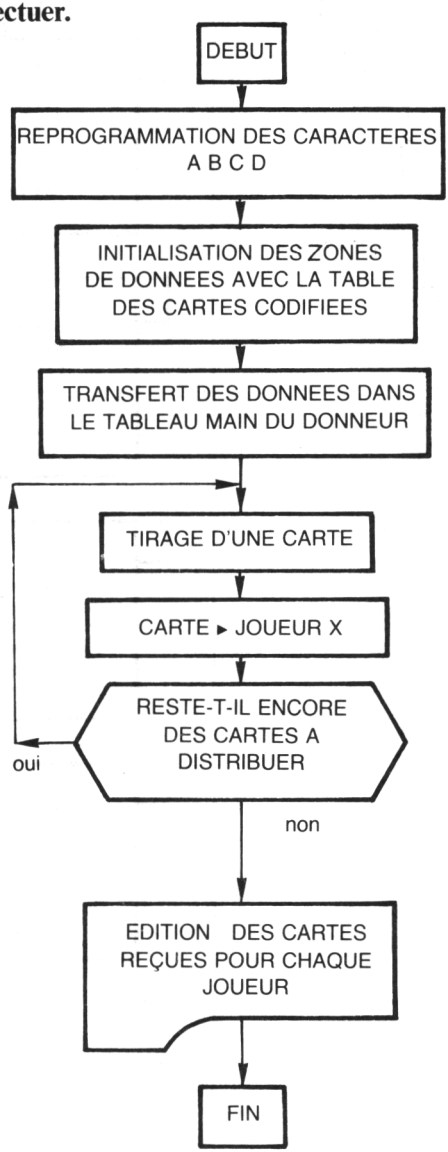
— **Présentation des résultats**

Nous afficherons pour un joueur le contenu du tableau de variables

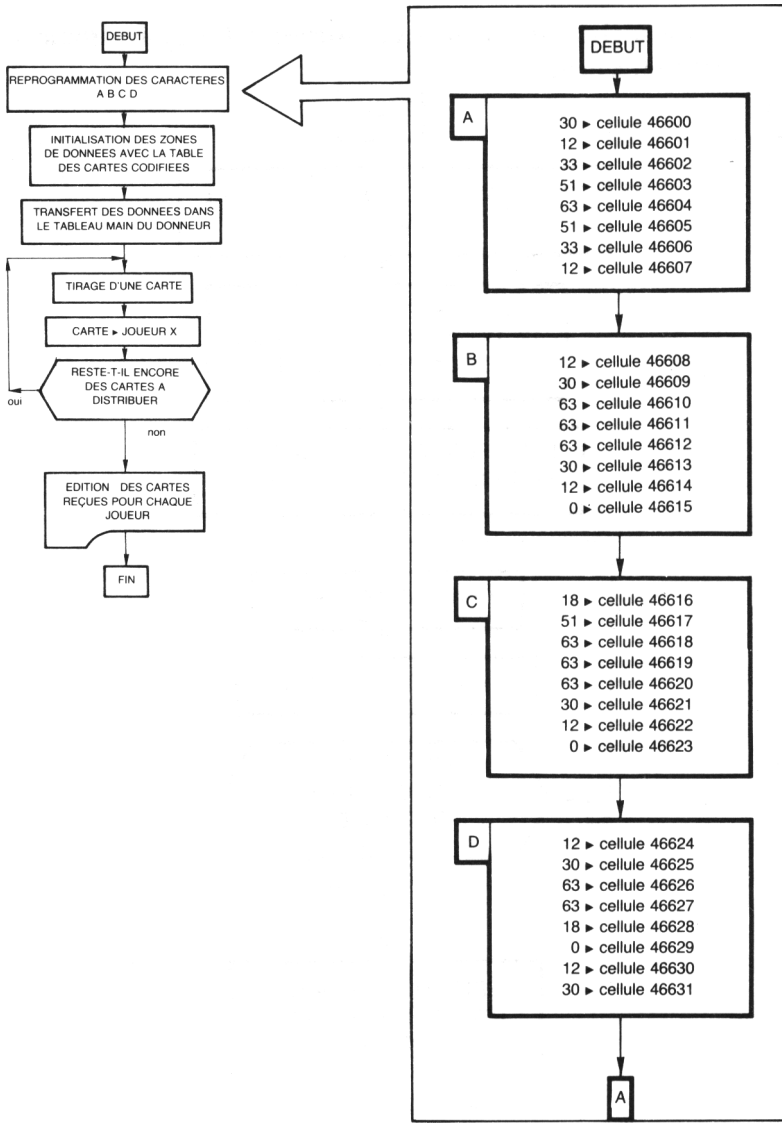
associées. Ce contenu sera le résultat de la distribution aléatoire issue de la main du donneur.

Nous répéterons cette opération pour les quatre joueurs.

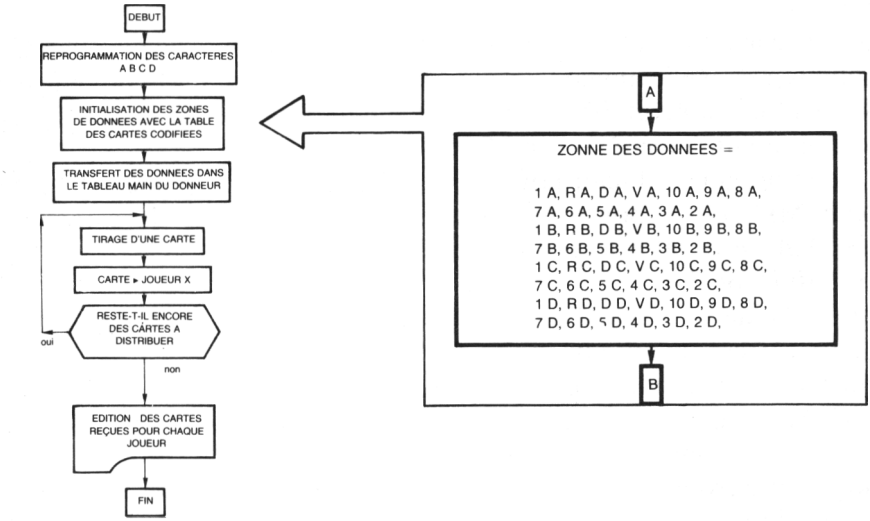
Détaillons cette analyse organique en construisant l'organigramme général du traitement à effectuer.



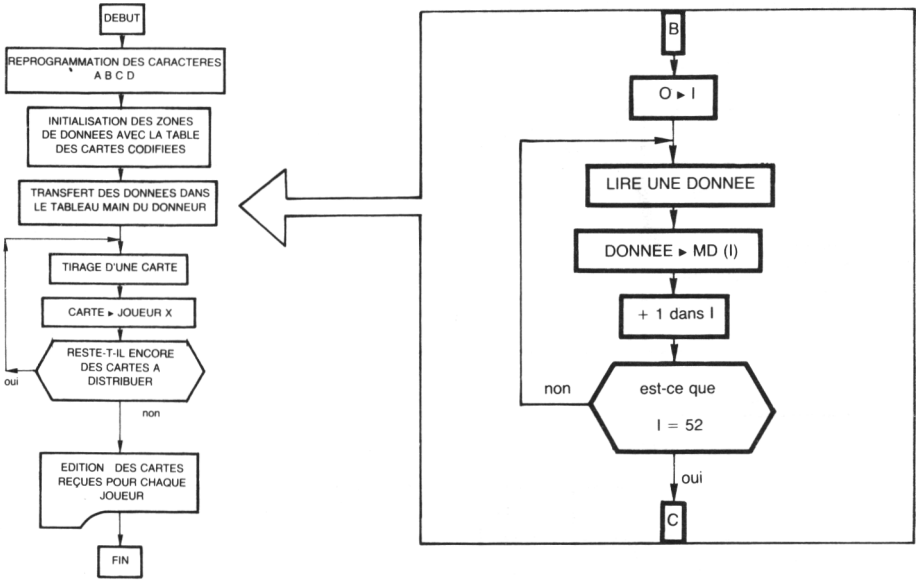
— Procédons à une analyse plus fine de chaque pavé de l'organigramme général.



Initialisation des zones de données de l'ordinateur par les valeurs codifiées représentant les 13 cartes par couleur.



Transfert des données dans le tableau « main du donneur ».
Nous choisirons de nommer ce tableau MD.



Distribution des cartes :

a) Extraction d'une carte au hasard dans le tableau MD (main du donneur).

Le tirage aléatoire d'un nombre ne doit plus vous poser de problèmes, de nombreuses explications ayant été données lors des exercices précédents.

Nous allons donc utiliser la même technique, mais, cette fois-ci, appliquée à un nombre compris entre 0 et 51.

Le nombre ainsi obtenu nous servira d'indice (ou de pointeur) sur le tableau MD. Nous pourrons donc extraire une chaîne de caractères codée à l'image de la carte qui aura été tirée.

Ex. : 10 D ou 9 B

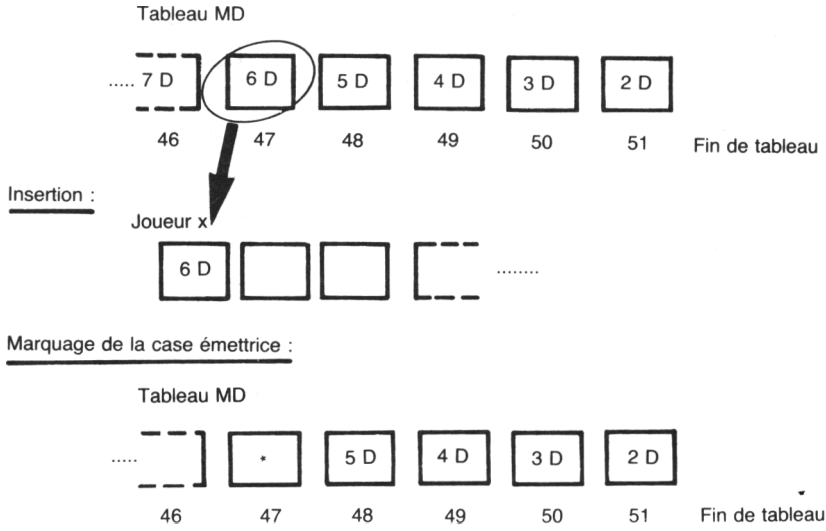
Cette chaîne de caractères sera ensuite insérée dans l'un des quatre tableaux correspondant aux mains des quatre joueurs.

Pour ne pas tirer plusieurs fois de suite la même carte, nous prendrons soin d'insérer dans la case émettrice une valeur indiquant que cette case a déjà été sélectionnée.

Nous répéterons cette opération jusqu'à épuisement des cartes du tableau MD.

En théorie tout a l'air parfaitement correct. Mais nous vous proposons d'observer ce qui se passera lors de l'exécution de cette stratégie.

Premier tirage : Nombre obtenu 47.



Deuxième tirage : Nombre obtenu 47. Effectivement, le nombre 47 a autant de chances de sortir que les 51 autres.

Pour les premiers tirages, le problème ne se pose pas, mais le tirage des dernières cartes risque de prendre beaucoup de temps. Effectivement, la probabilité de sélectionner des cases déjà tirées sera la même du début jusqu'à la fin. L'avant-dernière carte sera très longue à être sélectionnée.

Cette solution a l'avantage d'être simple à programmer. Essayons ensemble de réfléchir à une deuxième solution.

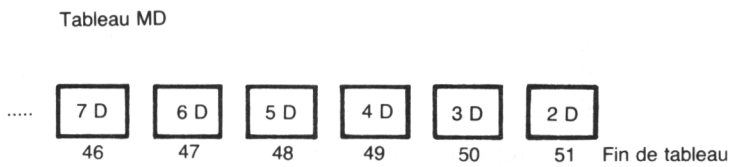
On obtiendra le nombre aléatoire de façon identique. Ce nombre sera l'indice de la case sélectionnée dans le tableau MD. Même procédé d'extraction et d'insertion dans le tableau du joueur x. Cette fois-ci, nous ne marquerons pas la case sélectionnée mais nous « retasserons » le tableau MD, à partir de la case extraite.

Evidemment cette solution aura l'inconvénient d'être plus compliquée à programmer, mais elle aura l'indiscutable avantage d'être courte lors de son exécution.

Attention : N'oublions pas, après le retassage du tableau MD, de procéder au tirage aléatoire portant sur un nombre de cases diminué de 1.

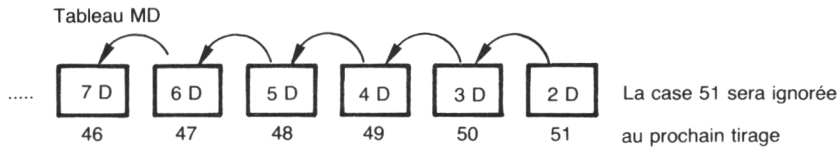
— Observons ce qui va se passer aux deux premiers coups.

Premier tirage : Tirage compris entre 0 et 51. Nombre obtenu : 46



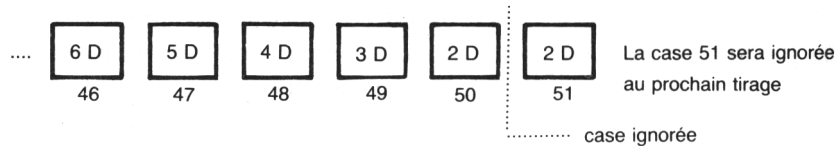
Insertion :

Retassage du tableau MD :



Deuxième tirage : Tirage compris entre 0 et 50. Nombre obtenu 48

Tableau MD



De proche en proche nous réduirons le tableau MD de 1, à chaque tirage. Nous procéderons ainsi à 51 tirages. La dernière carte sera extraite sans procéder au tirage d'un nombre compris entre 0 et 0 : la réponse est connue d'avance.

— Distribution de la carte

Quelques précisions sur les abréviations que nous allons utiliser.

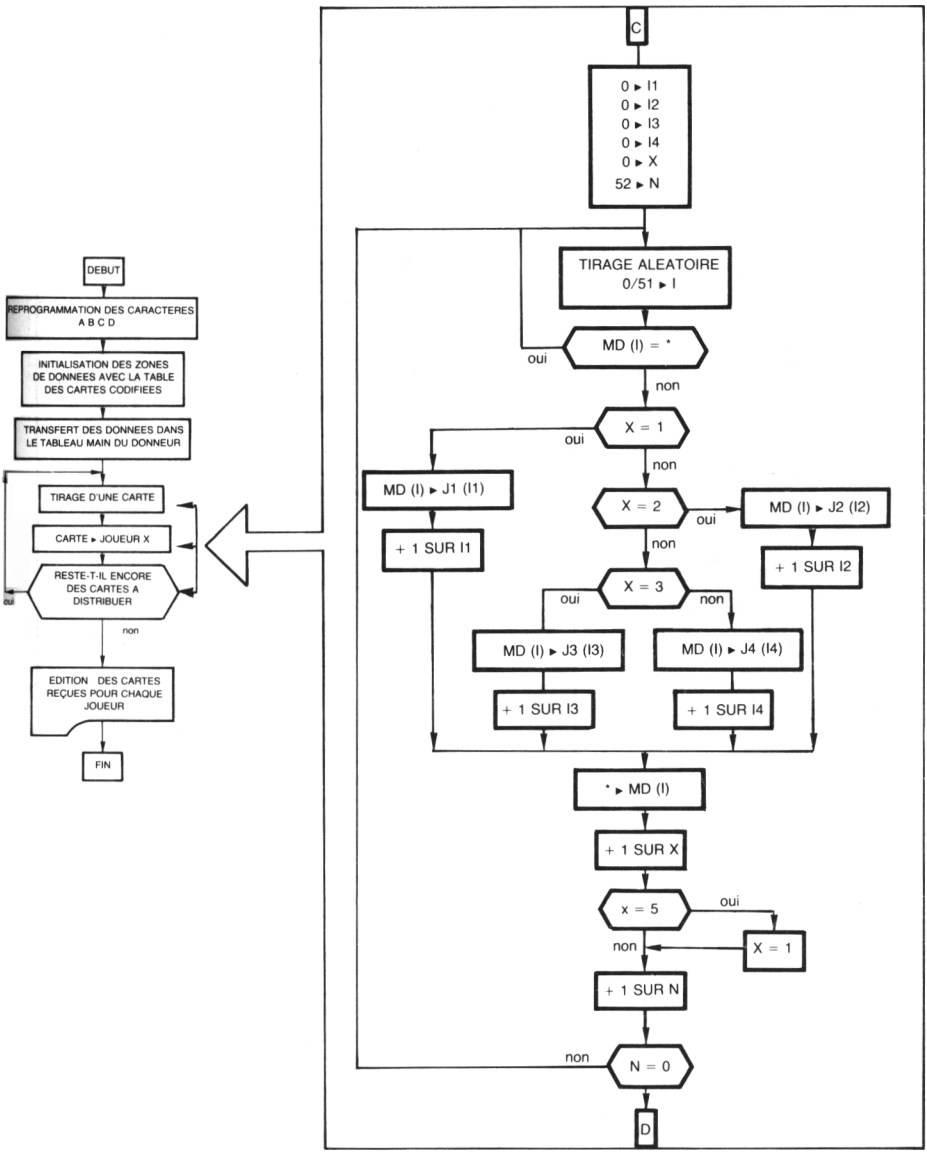
X - Numéro du joueur recevant la carte tirée
I1 - Indice de rangement dans le tableau du joueur 1
I2 - Indice de rangement dans le tableau du joueur 2
I3 - Indice de rangement dans le tableau du joueur 3
I4 - Indice de rangement dans le tableau du joueur 4

J1 - Tableau Joueur 1
J2 - Tableau Joueur 2
J3 - Tableau Joueur 3
J4 - Tableau Joueur 4

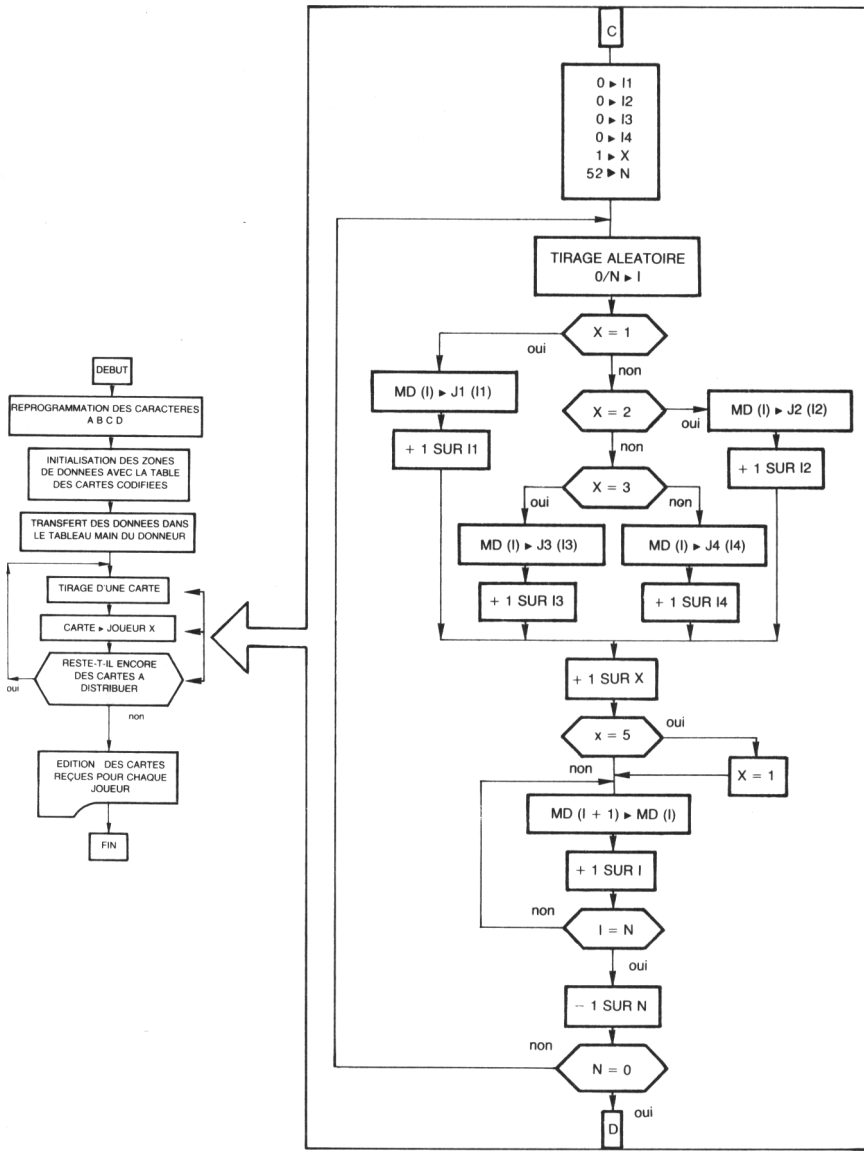
MD - Tableau main du donneur
I - Indice de tirage d'une case du tableau MD
N - Nombre de cartes restant à distribuer.

Les deux solutions proposées sont détaillées sur les pages suivantes.

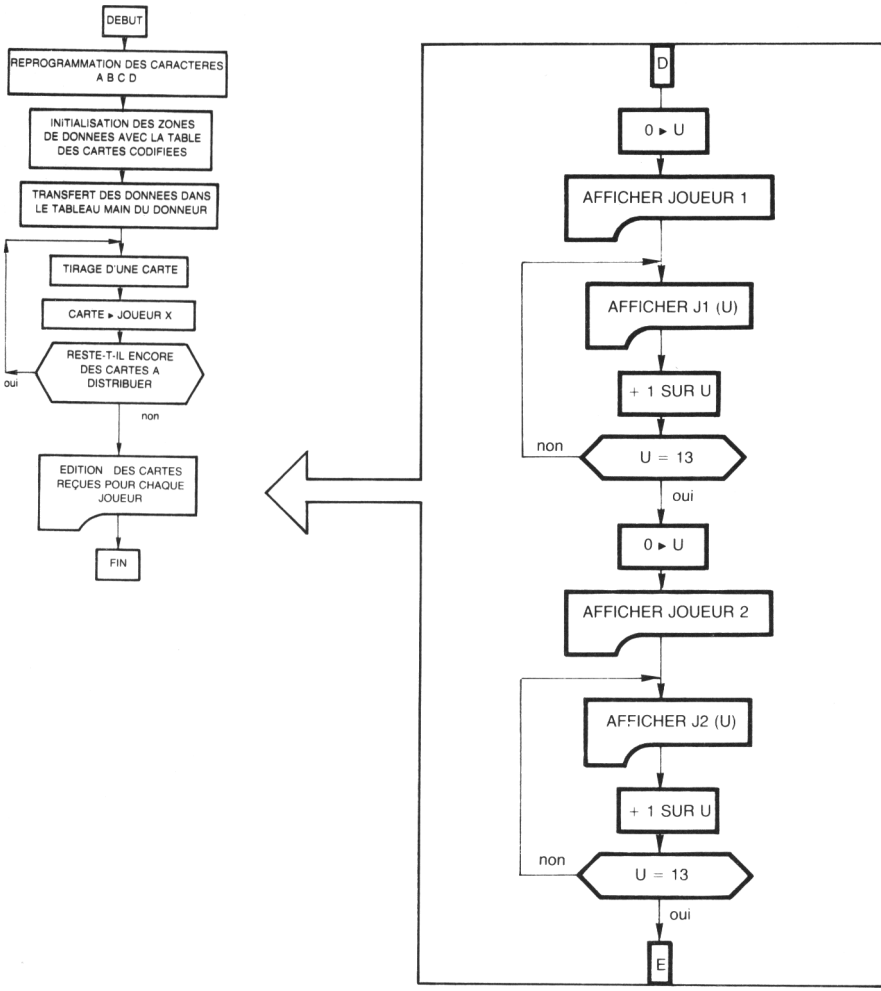
Solution 1 : Les cartes tirées sont remplacées par le caractère * (Astérisque).



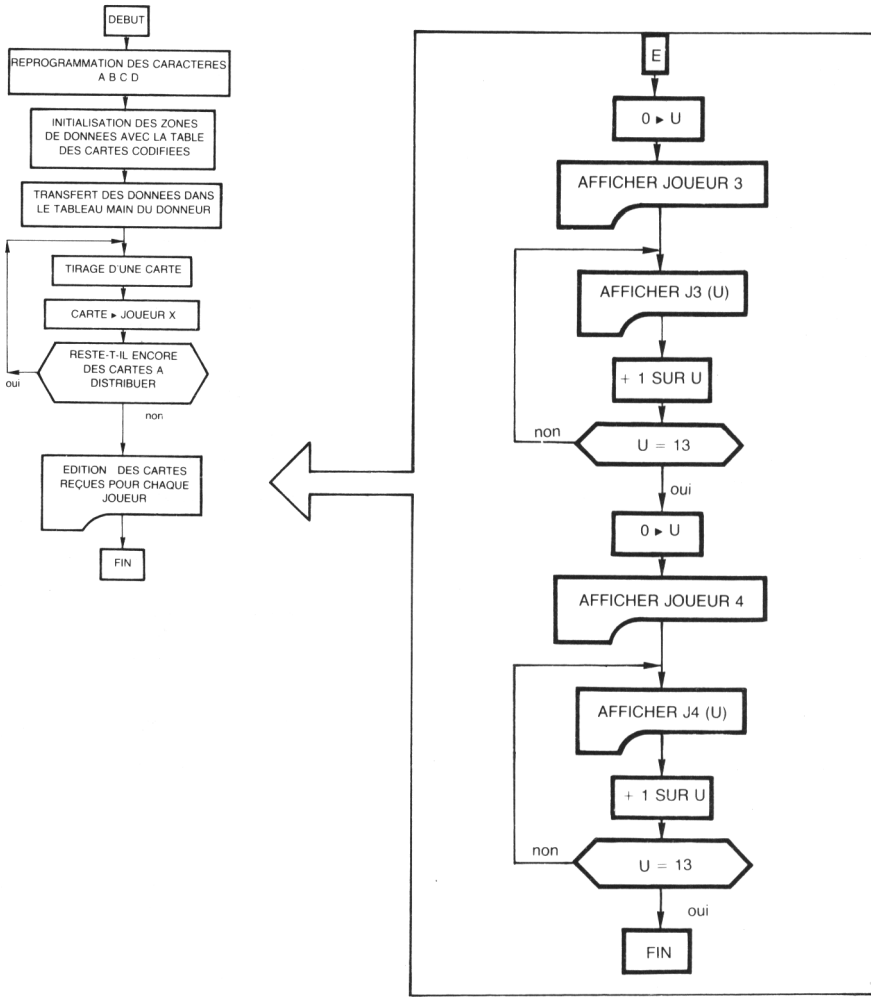
Solution 2 : Après chaque tirage de carte, le tableau MD est retassé.



Edition des résultats :



Edition des résultats (suite).



Solution 1

```
10 REM DISTRIBUTION AU BRIDGE
20 REM
30 REM REPROGRAMMATION DES CARACTERES
40 REM
50 CLS
60 REM
70 POKE 46600,30
80 POKE 46601,12
90 POKE 46602,33
100 POKE 46603,51
110 POKE 46604,63
120 POKE 46605,51
130 POKE 46606,33
140 POKE 46607,12
150 REM
160 REM CARCTERE .B. -> CARREAU
170 REM
180 POKE 46608,12
190 POKE 46609,30
200 POKE 46610,63
210 POKE 46611,63
220 POKE 46612,63
230 POKE 46613,30
240 POKE 46614,12
250 POKE 46615,0
260 REM
270 REM CARACTERE .C. -> COEUR
280 REM
290 POKE 46616,18
300 POKE 46617,51
310 POKE 46618,63
30 POKE 46619,63
330 POKE 46620,63
350 POKE 46621,30
360 POKE 46622,12
370 POKE 46623,0
380 REM
390 REM CARACTERE .D. -> PIOUE
400 REM
410 POKE 46624,12
420 POKE 46625,30
430 POKE 46626,63
440 POKE 46627,63
450 POKE 46628,18
460 POKE 46629,0
470 POKE 46630,12
480 POKE 46631,30
485 REM
490 REM PREPARATION DU JEU DE 52 CARTES
495 REM
499 GOTO 590
500 DATA 1 A,R A,Q A,V A,10 A,9 A,8 A,7 A,6 A,5 A,4 A,3 A,2 A
510 DATA 1 B,R B,Q B,V B,10 B,9 B,8 B,7 B,6 B,5 B,4 B,3 B,2 B
520 DATA 1 C,R C,Q C,V C,10 C,9 C,8 C,7 C,6 C,5 C,4 C,3 C,2 C
530 DATA 1 D,R D,Q D,V D,10 D,9 D,8 D,7 D,6 D,5 D,4 D,3 D,2 D
540 REM
```

```
550 REM INITIALISATION DU TABLEAU MD
560 REM
570 REM DIMENSIONNEMENT DES TABLEAUX
580 REM
590 DIM MD$(51)
600 DIM J1$(12):DIM J2$(12)
610 DIM J3$(12):DIM J4$(12)
620 FOR I=0 TO 51
630 READ MD$(I)
640 NEXT
650 REM
660 REM DISTRIBUTION DES CARTES
670 I1=0:I2=0:I3=0:I4=0
680 X=1:N=52
690 REM
700 REM TIRAGE ALEATOIRE DE L'INDICE
710 REM
715 REM    ** SOLUTION 2 **
720 I=INT(RND(1)*N)
730 REM
740 REM TEST SI CARTE DEJA DONNEE
750 REM
770 REM
780 REM DISTRIBUTION DE LA CARTE
790 REM
800 ON X GOTO 820,850,890,940
805 REM
810 REM JOUEUR 1
815 REM
820 J1$(I1) = MD$(I)
830 I1=I1+1
840 GOTO 1000
845 REM
850 REM JOUEUR 2
855 REM
860 J2$(I2) = MD$(I)
870 I2=I2+1
875 GOTO 1000
880 GOTO 1000
890 REM
895 REM JOUEUR 3
900 REM
910 J3$(I3) = MD$(I)
920 I3=I3+1
930 GOTO 1000
940 REM
945 REM JOUEUR 4
950 J4$(I4) = MD$(I)
960 I4=I4+1
985 REM
990 REM EFFACEMENT DE LA CARTE PAR *
995 REM
1000 MD$(I) = "*"
1010 IF X=5 THEN X=1
1020 REM
1030 REM RETASSEMENT DU TABLEAU MD
1040 REM
1050 MD$(I)=MD$(I+1)
```

```
1060 I=I+1
1070 IF I<N-1 THEN GOTO 1050
1080 REM
1090 REM RECHERCHE SUR N-1 CARTES
1100 REM
1110 N=N-1
1120 IF N>0 THEN GOTO 720
2000 REM
2005 REM  PRESENTATION DES JEUX DE
2010 REM  CHAQUE JOUEUR
2015 REM
2020 CLS
2030 PRINT"JOUEUR 1"
2040 FOR U=0 TO 12
2050 PRINTJ1$(U);"   ";
2060 NEXT
2070 PRINT:PRINT"JOUEUR 2"
2080 FOR U=0 TO 12
2090 PRINTJ2$(U);"   ";
2100 NEXT
2110 PRINT:PRINT"JOUEUR 3"
2120 FOR U=0 TO 12
2130 RINTJ3$(U);"   ";
2140 NEXT
2150 PRINT:PRINT"JOUEUR 4"
2160 FOR U=0 TO 12
2170 PRINTJ4$(U);"   ";
2180 NEXT
```

Solution 2

```
10 REM DISTRIBUTION AU BRIDGE
20 REM
30 REM REPROGRAMMATION DES CARACTERES
40 REM
50 CLS
60 REM
70 POKE 46600,30
80 POKE 46601,12
90 POKE 46602,33
100 POKE 46603,51
110 POKE 46604,63
120 POKE 46605,51
130 POKE 46606,33
140 POKE 46607,12
150 REM
160 REM CARACTERE .B. -> CARREAU
170 REM
180 POKE 46608,12
190 POKE 46609,30
200 POKE 46610,63
210 POKE 46611,63
220 POKE 46612,63
230 POKE 46613,30
240 POKE 46614,12
250 POKE 46615,0
```

```
260 REM
270 REM CARACTERE .C. -> COEUR
280 REM
290 POKE 46616,18
300 POKE 46617,51
310 POKE 46618,63
320 POKE 46619,63
330 POKE 46620,63
350 POKE 46621,30
360 POKE 46622,12
370 POKE 46623,0
380 REM
390 REM CARACTERE .D. -> PIQUE
400 REM
410 POKE 46624,12
420 POKE 46625,30
430 POKE 46626,63
440 POKE 46627,63
450 POKE 46628,18
460 POKE 46629,0
470 POKE 46630,12
480 POKE 46631,30
485 REM
490 REM PREPARATION DU JEU DE 52 CARTES
495 REM
499 GOTO 590
500 DATA 1 A,R A,Q A,V A,10 A,9 A,8 A,7 A,6 A,5 A,4 A,3 A,2 A
510 DATA 1 B,R B,Q B,V B,10 B,9 B,8 B,7 B,6 B,5 B,4 B,3 B,2 B
520 DATA 1 C,R C,Q C,V C,10 C,9 C,8 C,7 C,6 C,5 C,4 C,3 C,2 C
530 DATA 1 D,R D,Q D,V D,10 D,9 D,8 D,7 D,6 D,5 D,4 D,3 D,2 D
540 REM
550 REM INITIALISATION DU TABLEAU MD
560 REM
570 REM DIMENSIONNEMENT DES TABLEAUX
580 REM
590 DIM MD$(51)
600 DIM J1$(12):DIM J2$(12)
610 DIM J3$(12):IM J4$(12)
620 FOR I=0 TO 51
630 READ MD$(I)
640 NEXT
650 REM
660 REM DISTRIBUTION DES CARTES
670 I1=0:I2=0:I3=0:I4=0
680 X=1:N=52
690 REM
700 REM TIRAGE ALEATOIRE DE L'INDICE
710 REM
720 I=INT(RND(1)*52)
730 REM
740 REM TEST SI CARTE DEJA DONNEE
750 REM
760 IF MD$(I)="*" THEN GOTO 720
770 REM
780 REM DISTRIBUTION DE LA CARTE
790 REM
800 ON X GTO 820,850,90,940
805 REM
```

```
810 REM JOUEUR 1
815 REM
820 J1$(I1) = MD$(I)
830 I1=I1+1
840 GOTO 1000
845 REM
850 REM JOUEUR 2
855 REM
860 J2$(I2) = MD$(I)
870 I2=I2+1
875 GOTO 1000
880 GOTO 1000
890 REM
895 REM JOUEUR 3
900 REM
910 J3$(I3) = MD$(I)
920 I3=I3+1
930 GOTO 1000
940 REM
945 REM JOUEUR 4
950 J4$(I4) = MD$(I)
960 I4=I4+1
985 REM
990 REM EFFACEMENT DE LA CARTE PAR *
995 REM
1000 MD$(I) = "*"
1010 X=X+1
1020 IF X=5 THEN X=1
1030 N=N-1
1040 IF N>0 THEN GOTO 720
2000 REM
2005 REM  PRESENTATION DES JEUX DE
2010 REM  CHAQUE JOUEUR
2015 REM
2020 CLS
2030 PRINT"JOUEUR 1"
2040 FOR U=0 TO 12
2050 PRINTJ1$(U);"   ";
2060 NEXT
2070 PRINT:PRINT"JOUEUR 2"
2080 FOR U=0 TO 12
2090 PRINTJ2$(U);"   ";
2100 NEXT
2110 PRINT:PRINT"JOUEUR 3"
2120 FOR U=0 TO 12
2130 PRINTJ3$(U);"   ";
2140 NEXT
2150 PRINT:PRINT"JOUEUR 4"
2160 FOR U=0 TO 12
2170 PRINTJ4$(U);"   ";
2180 NEXT
```

CONCLUSIONS

Que pouvons-nous tirer comme enseignement.
D'une part :

LA PROGRAMMATION....

....C'EST LA SCIENCE DU DETAIL ET DE L'EXACTITUDE.

par ailleurs l'analyse — c'est-à-dire la formulation logique d'une solution informatique concrète — sera plus délicate.

PROGRAMMER C'EST FACILE....

....ANALYSER C'EST PLUS DIFFICILE.

De ces exercices se dégagent des points essentiels que vous devrez retenir, et qui seront toujours vérifiés quel que soit le niveau informatique que vous atteindrez.

1. Lire et posséder parfaitement le problème posé : il doit être précis, sans ambiguïté.

c'est **L'ANALYSE FONCTIONNELLE.**

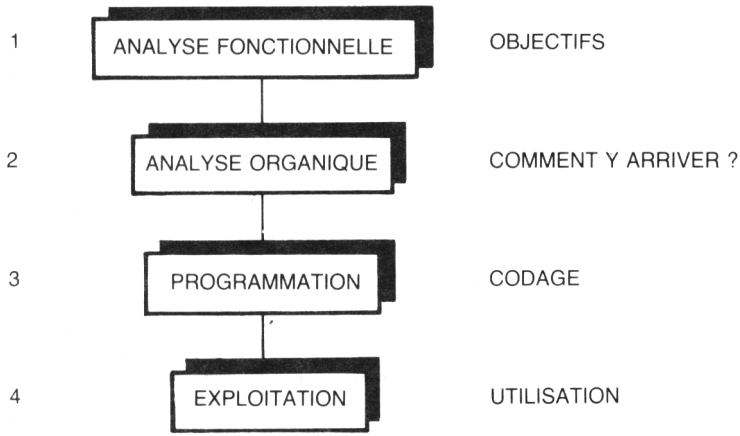
2. Vous devez être capable, après étude de l'analyse fonctionnelle, de réaliser la description fine et détaillée du travail de programmation

c'est **L'ANALYSE ORGANIQUE.**

C'est la partie la plus délicate à réaliser.

3. Si les conditions 1 et 2 sont réalisées, alors vous n'aurez aucun problème majeur pour l'écriture du programme. Ce programme ne sera que l'expression codée de l'analyse organique

c'est le travail de **PROGRAMMATION**



PROGRAMMER C'EST PREVOIR

CHAPITRE 7

La programmation structurée

7.1 AVANT-PROPOS

« En programmation, il ne suffit pas d'être inventif et ingénieux, il faut aussi savoir se discipliner et se contrôler pour ne pas être étouffé par ses propres complexités.

La nouvelle génération devra acquérir contrôle et discipline, principalement en apprenant à écrire des programmes correctement dès le début. Le processus de mise au point prendra une nouvelle forme : vérification qu'il n'y a pas d'erreurs, plutôt que l'ancienne forme où l'on détectait et corrigeait des erreurs, encore et encore (tester le programme par épuisement des cas possibles).

Programmer est un travail logique et sérieux, qui demande concentration et précision. Dans cette discipline, concentration et sécurité sont étroitement liées.

A titre d'illustration simple, considérez un enfant qui sait jouer parfaitement au jeu de tic-tac-toe, mais ne sait pas qu'il le sait. Si vous lui demandez de jouer pour quelque chose d'important, comme une sucette, il se dira « j'espère que je vais gagner ». Quelquefois il gagnera, quelquefois, il perdra. La seule raison pour laquelle il ne gagnera pas toujours est que son attention faiblira. Il ne s'en aperçoit pas parce qu'il considère sa victoire comme un hasard. Voyez comme les choses changent quand l'enfant « sait qu'il sait » jouer parfaitement. Il ne dira plus « j'espère que je vais gagner » mais au contraire « je sais que je vais gagner. C'est bon pour moi ». Et il verra la nécessité d'être attentif pour s'assurer la victoire.

En programmation comme au tic-tac-toe, il est caractéristique que l'attention va de pair avec une confiance justifiée dans ses propres capacités. Il ne suffit pas de savoir simplement comment écrire des programmes correctement. Le programmeur doit savoir qu'il sait le faire, et fournir l'attention correspondante ».

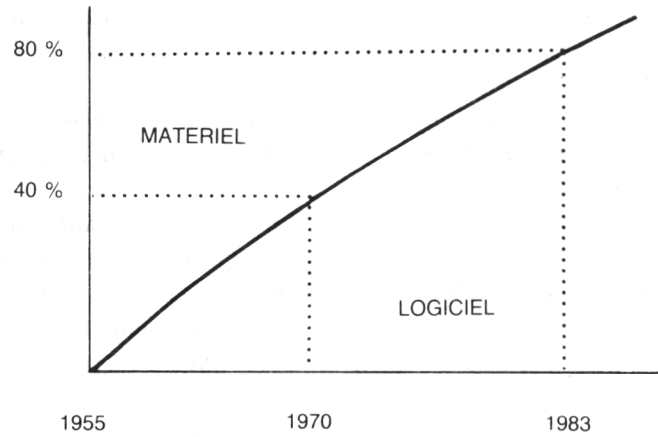
Harlan D. MILLS
Federal Systems Division, IBM
Gaithersburg, Maryland

7.2 SITUATION ACTUELLE

Avant de donner les préceptes fondamentaux de la programmation structurée, observons la situation actuelle de l'informatique.

LES COÛTS

Répartition matériels/logiciels



Les logiciels coûtent de plus en plus cher.

— Matériel ..... 20 %
— Réalisation des programmes ..... 30 %
— Maintenance des programmes ..... 50 %

La qualité des programmes est médiocre. Les coûts de développement et de maintenance deviennent exorbitants.

Plusieurs causes en sont à l'origine :

1. Méthodes de développement empiriques.
2. Manque d'organisation.
3. Mise au point délicate, voire impossible.
4. Maintenance coûteuse des programmes.

1.1 Méthodes de développement empiriques :

— Développement traditionnel.

On commence par le détail et on verra bien après.

C'est le « **BOTTOM-UP** »

— Absence d'algorithmes.
On essaye un bout de programme et on verra bien si ça marche.

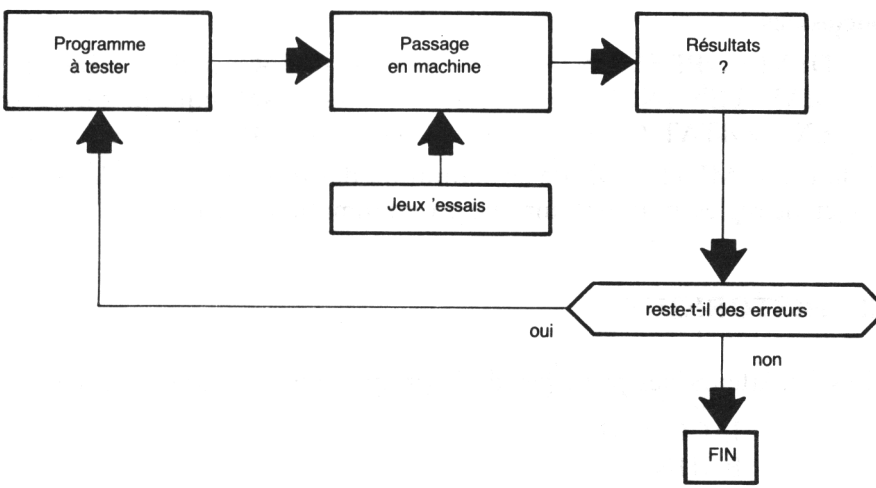
— Trop d'efforts pour réaliser un ensemble homogène.

— On ne se donne jamais le temps pour faire les choses bien mais on a toujours le temps de corriger ses erreurs. Nécessité fait loi.

2.1 Manque d'organisation :

Bien souvent, c'est le programmeur qui précise les spécifications fonctionnelles à mettre en œuvre. Il n'est pas toujours formé pour cela. On ne lui donne pas toujours les moyens et le temps d'assimilation nécessaire pour une bonne maîtrise du problème à traiter.

3.1 Mise au point délicate voire presque impossible :



Ce mode de fonctionnement risque de durer longtemps.

> LES ESSAIS NE SERVENT PAS A PROUVER QU'UN PROGRAMME EST CORRECT MAIS SEULEMENT QU'IL CONTIENT DES ERREURS
>
> **DIJKSTRA**

La mise au point nécessite 50 % du coût de développement d'une application.

4.1 Maintenance coûteuse des programmes :

— C'est le point faible des services informatiques et de tout programmeur.

— La correction des erreurs peut coûter jusqu'à cinquante fois plus cher que l'écriture du programme.

— Plus un programme vieillit, plus il est difficile de trouver les erreurs.

— Maintenance presque impossible pour un étranger.

— Absence de documentation, d'analyse.

— Commentaires inexistants dans les listings.

Conclusions.

— DEVELOPPEMENT ..... →Gaspillage d'énergie.

— MAINTENANCE ..... →Manque de souplesse.

— EXPLOITATION ..... →Peu efficace.

Il faut une METHODE de structuration des programmes, et le respect d'un jeu de règles, pour obtenir une programmation efficace.

7.3 HISTORIQUE

Les fondateurs des principes de la programmation structurée sont :

— JACOPINI
— BOHN
— DIJKSTRA
— MILLS
— KNUTH
— WIRTH

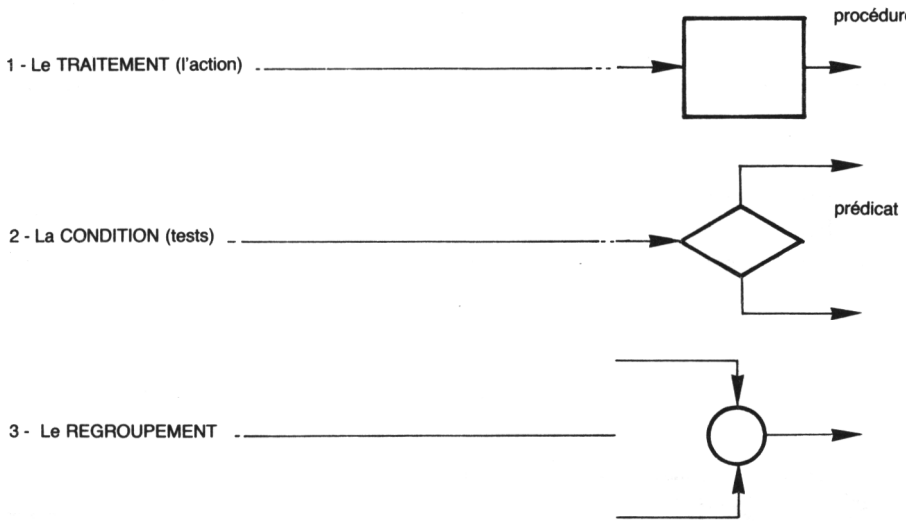
etc...

Caractéristiques :

— Modularité.
— Outils simples et efficaces.
— Décomposition des problèmes **TOP-DOWN** (du haut vers le détail).
— Plus de phase d'intégration.
— Compréhension par des non-informaticiens.
— Facilité de maintenance des programmes.

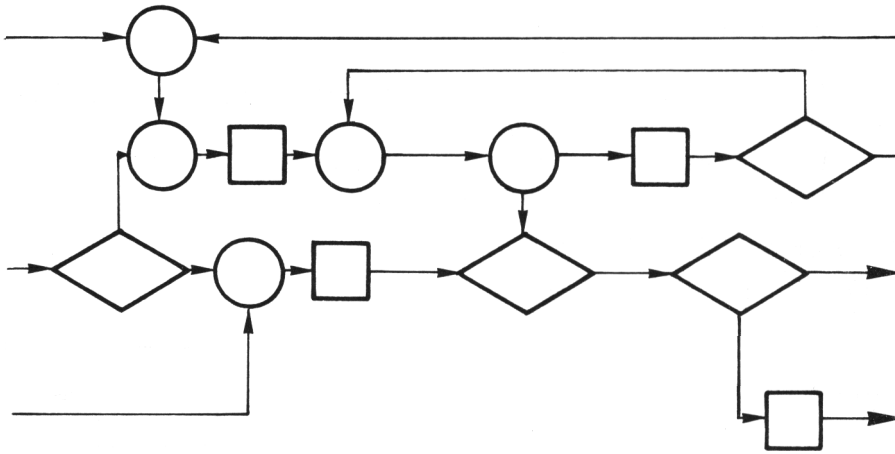
La programmation PROPRE :

On distingue TROIS constituants :



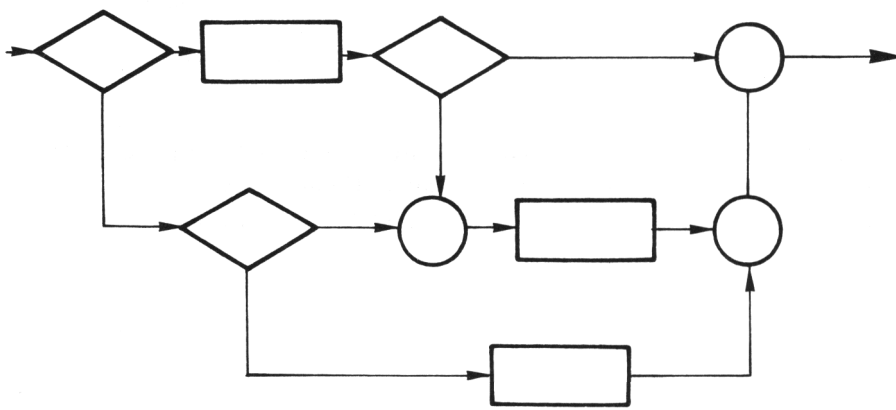
Un programme propre est bâti suivant le concept de la procédure.
Il n'aura **qu'une entrée** possible et **une sortie possible.**
Le chemin qui conduit de l'entrée à la sortie sera unique.

Programme non propre



— Plusieurs entrées.
— Plusieurs sorties.
— Tout sommet n'a pas un chemin vers la sortie.

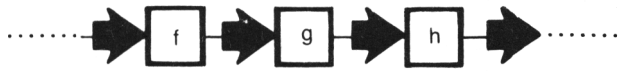
Programme propre



Une entrée. Une sortie.

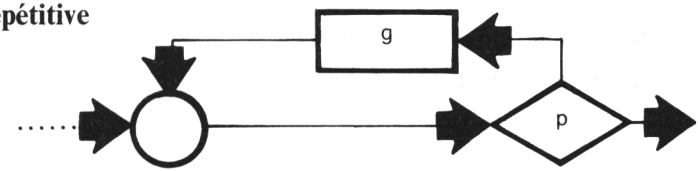
Les structures de bases :

1. L'enchaînement



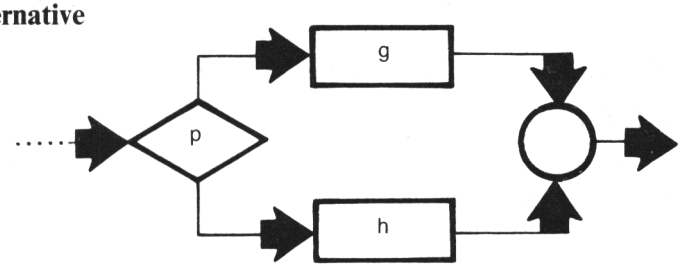

```
Ex. : 250 X = RX * SIN(AL) - 3
      251 Y = RY * COS(AL) - R
      260 Z = Z + 1
      270 V = 4
      .........
      etc...
```

2. La répétitive




```
Ex. : 100 REPEAT                    100 FOR I = 0 TO 10
      ............        ou        ....................
      ............                  ....................
      230 UNTIL I < > 10            230 NEXT I
```

3. L'alternative




```
Ex. : 500 IF X = 3 THEN
              ......
              ELSE
              ......
```

UN PROGRAMME STRUCTURÉ SERA TOUJOURS CONSTRUIT SUIVANT CES TROIS STRUCTURES

7.4 VINGT RÈGLES POUR BIEN PROGRAMMER

1. Définir complètement le problème

On a souvent tendance à travailler dans le vague.

Il manque la plupart du temps les spécifications exactes du travail à réaliser.

L'art de programmer, c'est savoir apprécier la pertinence des énoncés.

2. Réfléchir avant d'agir

Talleyrand disait : « Quand je suis pressé, je prends mon temps ».

Faites la même chose en programmation. Vous atteindrez plus rapidement le but fixé.

3. Utiliser la décomposition TOP-DOWN

C'est la règle fondamentale de la programmation structurée.

Elle permet ensuite de développer un programme dans n'importe quel langage.

Il est préférable de concevoir une analyse « portable » que de prévoir la portabilité des langages.

4. Construire logiquement

Il faut structurer les programmes en modules simples.

Il faut éviter que certaines spécifications soient partagées dans plusieurs modules.

C'est la PROGRAMMATION MODULAIRE.

5. Eviter le GOTO

Ce qui n'est pas facile...

Il existe même des langages qui n'en ont pas (PASCAL).

Les structures de bases le permettent.

> **« LA QUALITE D'UN PROGRAMME EST INVERSEMENT PROPORTIONNELLE AU NOMBRE DE GOTO»**
>
> DIJKSTRA

L'usage généreux du GOTO conduit obligatoirement à une programmation dite **SPAGHETTI.** On ne sait pas où cela commence et où cela finit.

6. Soigner la syntaxe

En dehors des erreurs de frappe, les fautes d'écritures sur la syntaxe ne sont pas pardonnables.

Il faut éviter au maximum les ennuis, les tests en donnent assez.

7. Bien choisir le nom des variables

Utiliser des noms qui soient en rapport avec la fonction.

Il faut adopter tout de suite une méthode MNEMOTECHNIQUE.

8. Bien utiliser les variables intermédiaires

Eviter les instructions complexes et bourrées d'astuces.

Faire un usage judicieux des noms de variables.

9. Ne pas toucher aux indices de boucles

10. Eviter de répéter des calculs pour rien

Le temps est précieux sur un ordinateur.

11. Programmation sans astuces

Un usage trop abusif des astuces donne une structure médiocre, un manque de clarté et finalement un manque d'efficacité.

Eviter le perfectionnisme.

12. Prévoir des aides à la mise au point

Il n'existe pas de programme qui tourne au premier essai.

Prévoir des impressions ou affichages intermédiaires, des arrêts de contrôle. Il sera très facile de les retirer quand tout sera au point.

13. L'ordinateur ne pense pas à votre place

L'imprécision conduit à l'imprévisible.

14. Commenter les programmes

C'est le mode d'emploi.

Les commentaires ont une valeur inestimable et décrivent la structure logique de l'algorithme.

15. Soigner la présentation

C'est le plus facile à faire.

C'est plus lisible et plus agréable pour les autres.

16. Respecter les conventions du langage

On est mieux compris par les autres.

17. Tester les programmes à la main

Evidemment, on est très pressé de voir le résultat mais on est toujours déçu quand ça ne fonctionne pas après plusieurs essais.

Les tests à la main évitent ces déceptions et font gagner du temps.

18. Ne pas s'occuper de la présentation tant que le programme ne tourne pas

19. Soigner la présentation quand le programme tourne

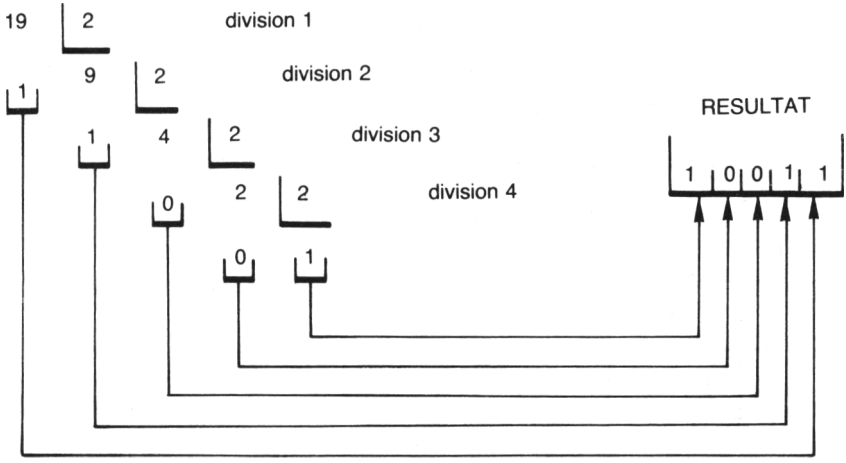
20. Ne pas avoir peur de tout recommencer

Annexes

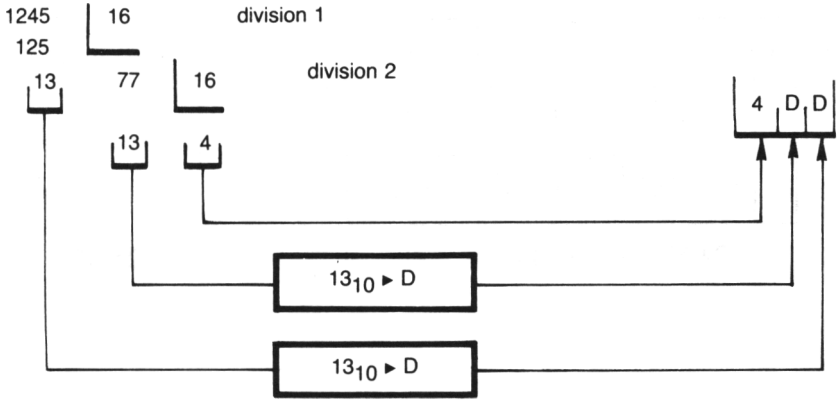
ANNEXE A CONVERSION D'UN NOMBRE DÉCIMAL VERS UNE BASE « r »

Convertir un nombre, c'est diviser celui-ci par la base « r ». Les divisions s'arrêteront dès que le résultat d'une division sera inférieur à la base « r ». Le résultat final de la conversion est le rapprochement des restes successifs de chaque division et s'écrit à gauche de chaque reste précédent écrit. Le résultat de la dernière division sera le dernier chiffre placé à gauche des chiffres précédemment obtenus.

Ex. 1 : convertir 19 en base 2 (Binaire)



Ex. 2 : convertir 1245 en base 16 (Hexadécimal)



ANNEXE B CONVERSION D'UN NOMBRE D'UN SYSTÈME DE BASE « r » VERS LE DÉCIMAL

Le résultat exprimé en décimal sera l'addition des « n » produits : chiffre de rang n par base de numération élevée à la puissance du rang correspondant.

Ex. 1 : $1\,847_{10}$ à convertir en décimal (ne sert qu'à la vérification)

rangs	4	3	2	1	0	puissances de 10
		1 1×10^3	8 8×10^2	4 4×10^1	7 7×10^0	

$1\,000 + 800 + 40 + 7 = \mathbf{1\,847}$

Ex. 2 : $1\,010_2$ à convertir en décimal

rangs	4	3	2	1	0	puissances de 2
		1	0	1	0	
		1×2^3	0×2^2	1×2^1	0×2^0	

$8 + 0 + 2 + 0 = \mathbf{10}$

Ex. 3 : $6F3A_{16}$ à convertir en décimal

rangs	4	3	2	1	0	puissances de 16
		6	F	3	A	
		6×16^3	15×16^2	3×16^1	10×16^0	

$24\,576 + 3\,840 + 48 + 10 = \mathbf{28\,474}$

ANNEXE C TABLEAU DE CONVERSION DÉCIMAL-BINAIRE DES 256 PREMIERS ENTIERS POSITIFS

0	0000 0000	64	0100 0000	128	1000 0000	192	1100 0000
1	0000 0001	65	0100 0001	129	1000 0001	193	1100 0001
2	0000 0010	66	0100 0010	130	1000 0010	194	1100 0010
3	0000 0011	67	0100 0011	131	1000 0011	195	1100 0011
4	0000 0100	68	0100 0100	132	1000 0100	196	1100 0100
5	0000 0101	69	0100 0101	133	1000 0101	197	1100 0101
6	0000 0110	70	0100 0110	134	1000 0110	198	1100 0110
7	0000 0111	71	0100 0111	135	1000 0111	199	1100 0111
8	0000 1000	72	0100 1000	136	1000 1000	200	1100 1000
9	0000 1001	73	0100 1001	137	1000 1001	201	1100 1001
10	0000 1010	74	0100 1010	138	1000 1010	202	1100 1010
11	0000 1011	75	0100 1011	139	1000 1011	203	1100 1011
12	0000 1100	76	0100 1100	140	1000 1100	204	1100 1100
13	0000 1101	77	0100 1101	141	1000 1101	205	1100 1101
14	0000 1110	78	0100 1110	142	1000 1110	206	1100 1110
15	0000 1111	79	0100 1111	143	1000 1111	207	1100 1111
16	0001 0000	80	0101 0000	144	1001 0000	208	1101 0000
17	0001 0001	81	0101 0001	145	1001 0001	209	1101 0001
18	0001 0010	82	0101 0010	146	1001 0010	210	1101 0010
19	0001 0011	83	0101 0011	147	1001 0011	211	1101 0011
20	0001 0100	84	0101 0100	148	1001 0100	212	1101 0100
21	0001 0101	85	0101 0101	149	1001 0101	213	1101 0101
22	0001 0110	86	0101 0110	150	1001 0110	214	1101 0110
23	0001 0111	87	0101 0111	151	1001 0111	215	1101 0111
24	0001 1000	88	0101 1000	152	1001 1000	216	1101 1000
25	0001 1001	89	0101 1001	153	1001 1001	217	1101 1001
26	0001 1010	90	0101 1010	154	1001 1010	218	1101 1010
27	0001 1011	91	0101 1011	155	1001 1011	219	1101 1011
28	0001 1100	92	0101 1100	156	1001 1100	220	1101 1100
29	0001 1101	93	0101 1101	157	1001 1101	221	1101 1101
30	0001 1110	94	0101 1110	158	1001 1110	222	1101 1110
31	0001 1111	95	0101 1111	159	1001 1111	223	1101 1111
32	0010 0000	96	0110 0000	160	1010 0000	224	1110 0000

ANNEXE C *(suite)*

33	0010	0001	97	0110	0001	161	1010	0001	225	1110	0001
34	0010	0010	98	0110	0010	162	1010	0010	226	1110	0010
35	0010	0011	99	0110	0011	163	1010	0011	227	1110	0011
36	0010	0100	100	0110	0100	164	1010	0100	228	1110	0100
37	0010	0101	101	0110	0101	165	1010	0101	229	1110	0101
38	0010	0110	102	0110	0110	166	1010	0110	230	1110	0110
39	0010	0111	103	0110	0111	167	1010	0111	231	1110	0111
40	0010	1000	104	0110	1000	168	1010	1000	232	1110	1000
41	0010	1001	105	0110	1001	169	1010	1001	233	1110	1001
42	0010	1010	106	0110	1010	170	1010	1010	234	1110	1010
43	0010	1011	107	0110	1011	171	1010	1011	235	1110	1011
44	0010	1100	108	0110	1100	172	1010	1100	236	1110	1100
45	0010	1101	109	0110	1101	173	1010	1101	237	1110	1101
46	0010	1110	110	0110	1110	174	1010	1110	238	1110	1110
47	0010	1111	111	0110	1111	175	1010	1111	239	1110	1111
48	0011	0000	112	0111	0000	176	1011	0000	240	1111	0000
49	0011	0001	113	0111	0001	177	1011	0001	241	1111	0001
50	0011	0010	114	0111	0010	178	1011	0010	242	1111	0010
51	0011	0011	115	0111	0011	179	1011	0011	243	1111	0011
52	0011	0100	116	0111	0100	180	1011	0100	244	1111	0100
53	0011	0101	117	0111	0101	181	1011	0101	245	1111	0101
54	0011	0110	118	0111	0110	182	1011	0110	246	1111	0110
55	0011	0111	119	0111	0111	183	1011	0111	247	1111	0111
56	0011	1000	120	0111	1000	184	1011	1000	248	1111	1000
57	0011	1001	121	0111	1001	185	1011	1001	249	1111	1001
58	0011	1010	122	0111	1010	186	1011	1010	250	1111	1010
59	0011	1011	123	0111	1011	187	1011	1011	251	1111	1011
60	0011	1100	124	0111	1100	188	1011	1100	252	1111	1100
61	0011	1101	125	0111	1101	189	1011	1101	253	1111	1101
62	0011	1110	126	0111	1110	190	1011	1110	254	1111	1110
63	0011	1111	127	0111	1111	191	1011	1111	255	1111	1111

ANNEXE D TABLEAU DE CONVERSION DÉCIMAL-HEXADÉCIMAL DES 512 PREMIERS ENTIERS POSITIFS

0	0	64	40	128	80	192	C0	256	100	320	140	384	180	448	1C0
1	1	65	41	129	81	193	C1	257	101	321	141	385	181	449	1C1
2	2	66	42	130	82	194	C2	258	102	322	142	386	182	450	1C2
3	3	67	43	131	83	195	C3	259	103	323	143	387	183	451	1C3
4	4	68	44	132	84	196	C4	260	104	324	144	388	184	452	1C4
5	5	69	45	133	85	197	C5	261	105	325	145	389	185	453	1C5
6	6	70	46	134	86	198	C6	262	106	326	146	390	186	454	1C6
7	7	71	47	135	87	199	C7	263	107	327	147	391	187	455	1C7
8	8	72	48	136	88	200	C8	264	108	328	148	392	188	456	1C8
9	9	73	49	137	89	201	C9	265	109	329	149	393	189	457	1C9
10	A	74	4A	138	8A	202	CA	266	10A	330	14A	394	18A	458	1CA
11	B	75	4B	139	8B	203	CB	267	10B	331	14B	395	18B	459	1CB
12	C	76	4C	140	8C	204	CC	268	10C	332	14C	396	18C	460	1CC
13	D	77	4D	141	8D	205	CD	269	10D	333	14D	397	18D	461	1CD
14	E	78	4E	142	8E	206	CE	270	10E	334	14E	398	18E	462	1CE
15	F	79	4F	143	8F	207	CF	271	10F	335	14F	399	18F	463	1CF
16	10	80	50	144	90	208	D0	272	110	336	150	400	190	464	1D0
17	11	81	51	145	91	209	D1	273	111	337	151	401	191	465	1D1
18	12	82	52	146	92	210	D2	274	112	338	152	402	192	466	1D2
19	13	83	53	147	93	211	D3	275	113	339	153	403	193	467	1D3
20	14	84	54	148	94	212	D4	276	114	340	154	404	194	468	1D4
21	15	85	55	149	95	213	D5	277	115	341	155	405	195	469	1D5
22	16	86	56	150	96	214	D6	278	116	342	156	406	196	470	1D6
23	17	87	57	151	97	215	D7	279	117	343	157	407	197	471	1D7
24	18	88	58	152	98	216	D8	280	118	344	158	408	198	472	1D8
25	19	89	59	153	99	217	D9	281	119	345	159	409	199	473	1D9
26	1A	90	5A	154	9A	218	DA	282	11A	346	15A	410	19A	474	1DA
27	1B	91	5B	155	9B	219	DB	283	11B	347	15B	411	19B	475	1DB
28	1C	92	5C	156	9C	220	DC	284	11C	348	15C	412	19C	476	1DC
29	1D	93	5D	157	9D	221	DD	285	11D	349	15D	413	19D	477	1DD
30	1E	94	5E	158	9E	222	DE	286	11E	350	15E	414	19E	478	1DE
31	1F	95	5F	159	9F	223	DF	287	11F	351	15F	415	19F	479	1DF
32	20	96	60	160	A0	224	E0	288	120	352	160	416	1A0	480	1E0
33	21	97	61	161	A1	225	E1	289	121	353	161	417	1A1	481	1E1
34	22	98	62	162	A2	226	E2	290	122	354	162	418	1A2	482	1E2
35	23	99	63	163	A3	227	E3	291	123	355	163	419	1A3	483	1E3
36	24	100	64	164	A4	228	E4	292	124	356	164	420	1A4	484	1E4
37	25	101	65	165	A5	229	E5	293	125	357	165	421	1A5	485	1E5
38	26	102	66	166	A6	230	E6	294	126	358	166	422	1A6	486	1E6
39	27	103	67	167	A7	231	E7	295	127	359	167	423	1A7	487	1E7
40	28	104	68	168	A8	232	E8	296	128	360	168	424	1A8	488	1E8
41	29	105	69	169	A9	233	E9	297	129	361	169	425	1A9	489	1E9
42	2A	106	6A	170	AA	234	EA	298	12A	362	16A	426	1AA	490	1EA
43	2B	107	6B	171	AB	235	EB	299	12B	363	16B	427	1AB	491	1EB

ANNEXE D *(suite)*

44	2C	108	6C	172	AC	236	EC	300	12C	364	16C	428	1AC	492	1EC
45	2D	109	6D	173	AD	237	ED	301	12D	365	16D	429	1AD	493	1ED
46	2E	110	6E	174	AE	238	EE	302	12E	366	16E	430	1AE	494	1EE
47	2F	111	6F	175	AF	239	EF	303	12F	367	16F	431	1AF	495	1EF
48	30	112	70	176	B0	240	F0	304	130	368	170	432	1B0	496	1F0
49	31	113	71	177	B1	241	F1	305	131	369	171	433	1B1	497	1F1
50	32	114	72	178	B2	242	F2	306	132	370	172	434	1B2	498	1F2
51	33	115	73	179	B3	243	F3	307	133	371	173	435	1B3	499	1F3
52	34	116	74	180	B4	244	F4	308	134	372	174	436	1B4	500	1F4
53	35	117	75	181	B5	245	F5	309	135	373	175	437	1B5	501	1F5
54	36	118	76	182	B6	246	F6	310	136	374	176	438	1B6	502	1F6
55	37	119	77	183	B7	247	F7	311	137	375	177	439	1B7	503	1F7
56	38	120	78	184	B8	248	F8	312	138	376	178	440	1B8	504	1F8
57	39	121	79	185	B9	249	F9	313	139	377	179	441	1B9	505	1F9
58	3A	122	7A	186	BA	250	FA	314	13A	378	17A	442	1BA	506	1FA
59	3B	123	7B	187	BB	251	FB	315	13B	379	17B	443	1BB	507	1FB
60	3C	124	7C	188	BC	252	FC	316	13C	380	17C	444	1BC	508	1FC
61	3D	125	7D	189	BD	253	FD	317	13D	381	17D	445	1BD	509	1FD
62	3E	126	7E	190	BE	254	FE	318	13E	382	17E	446	1BE	510	1FE
63	3F	127	7F	191	BF	255	FF	319	13F	383	17F	447	1BF	511	1FF

ANNEXE E CODE ASCII

Caractères	Valeurs Dec.	Valeurs Hexa.	Caractères	Valeurs Dec.	Valeurs Hexa.
espace	32	20	P	80	50
!	33	21	Q	81	51
"	34	22	R	82	52
#	35	23	S	83	53
$	36	24	T	84	54
%	37	25	U	85	55
&	38	26	V	86	56
'	39	27	W	87	57
(	40	28	X	88	58
)	41	29	Y	89	59
*	42	2A	Z	90	5A
+	43	2B	[	91	5B
,	44	2C	\	92	5C
–	45	2D	]	93	5D
.	46	2E	↑	94	5E
/	47	2F	£	95	5F
0	48	30	©	96	60
1	49	31	a	97	61
2	50	32	b	98	62
3	51	33	c	99	63
4	52	34	d	100	64
5	53	35	e	101	65
6	54	36	f	102	66
7	55	37	g	103	67
8	56	38	h	104	68
9	57	39	i	105	69
:	58	3A	j	106	6A
;	59	3B	k	107	6B
<	60	3C	l	108	6C
=	61	3D	m	109	6D
>	62	3E	n	110	6E
?	63	3F	o	111	6F
@	64	40	p	112	70

ANNEXE E *(suite)*

A	65	41	q	113	71
B	66	42	r	114	72
C	67	43	s	115	73
D	68	44	t	116	74
E	69	45	u	117	75
F	70	46	v	118	76
G	71	47	w	119	77
H	72	48	x	120	78
I	73	49	y	121	79
J	74	4A	z	122	7A
K	75	4B	{	123	7B
L	76	4C	|	124	7C
M	77	4D	}	125	7D
N	78	4E			
O	79	4F			

ANNEXE F RAPPEL DES PRINCIPALES INSTRUCTIONS DU LANGAGE BASIC

Variables simples :

Type	nom	étendue
Réel	AA	+/− 9.99999999E+37
Entier	AA%	+/− 32767
Chaîne	AA$	0 à 255 caractères

Variables tableaux :

Tout type possible.
Dimension limitée par la mémoire

Opérateurs algébriques :

=	Assignation (LET option.)
^	Elévation à une puissance
*	Multiplication
/	Division
−	Soustraction
+	Addition

Opérateurs relationnels et logiques :

=	Egalité
< > ou > <	Différent
<	Inférieur
< =	inférieur ou égal
>	supérieur
> =	supérieur ou égal
NOT	NON logique
AND	ET logique
OR	OU logique
FALSE	Booléen. Vaut 0 (Faux)
TRUE	Booléen. Vaut − 1 (Vrai)

Les expressions logiques prennent la valeur − 1 si vraie et 0 si fausse. Elles peuvent comparer des chaînes de caractères.

Tableaux et chaînes :

DIM A(X,Y,Z)	Donne la dimension maxi du tableau A. Z grilles de X lignes, Y colonnes.
DIM A$(..	Chaque élément peut contenir 1 chaîne de 255 caractères maxi.
LEN (A$)	Donne le nombre de caractères contenus dans la variable A$
STR$(Z)	Convertit une valeur Z en une chaîne de caractères.
VAL(A$)	Convertit la chaîne contenue dans A$ en une valeur numérique.
CHR$(X)	Donne le caractère dont le code ASCII est contenu dans X
ASC(A$)	Donne le code ASCII du premier caractère contenu dans A$.
LEFT$(A$,X)	Donne les X caractères de gauche contenus dans A$
RIGHT$(A$,X)	Donne les X caractères de droite contenus dans A$
MID$(A$,X,Y)	Donne les Y caractères de A$ à partir de la position X
B$ = A$ + ABDC	+ concatène des chaînes de caractères.
HEX$...	Valeur hexadécimale.

Les commandes :

LIST	Affiche le programme à l'écran
LIST 50	Affiche la ligne 50
LIST-50	Affiche du début jusqu;à la ligne 50
LIST 50-	Affiche de la ligne 50 jusqu'à la fin
LIST50-200	Affiche de la ligne 50 à 200.
NEW	Efface le programme en mémoire
RUN	Lance l'exécution du programme en mémoire
RUN 200	Lance l'exécution du programme en mémoire à partir de la ligne 200.
(CTRL) C	Arrêt de l'exécution d'un programme. Arrêt de l'affichage.
espace	Arrêt temporaire d'une liste.
CONT	Relance d'un programme après (CTRL)C ou STOP.

CLEAR Remet à zéro toutes les variables et vide toutes les chaînes.

HIMEM #8000 Positionne l'adresse la plus haute de mémoire utilisable pour un programme Basic.

Les instructions d'entrée/sortie :

INPUT A/A$/A% Attend que l'utilisateur compose une saisie au clavier validée par RETURN. Résultats transférés dans variables.

GET A$ Attend qu'une touche soit enfoncée, puis reprend l'exécution du programme.

X$=KEY$ Lit le clavier. Si une touche est enfoncée alors il transfert sa valeur dans X$ puis continue l'exécution du progr. Si rien il ne s'arrête pas et continue l'exécution.

PRINT A/A$/A% Affiche à l'écran le contenu des variables : réel, chaîne ou entier.

PRINT"ABCDE" Affiche la chaîne entre " ".

Les instructions relatives au déroulement du programme

IF cond. Permet de comparer des variables suivant un ou plusieurs opérateurs relationnels.

THEN inst. Répond à la condition vraie après un IF.

ELSE inst. Répond à la condition fausse après un IF

... Ces trois ordres sont utilisés ensemble.

GOTO 400 Ordre de branchement inconditionnel à la ligne 400.

ON X GOTO 100,200.. Ordre de branchements multiples.
X=1 GOTO 100,
X=2 GOTO 200...

GOSUB 400 Ordre de branchement inconditionnel à la ligne 400. Retour après GOSUB par RETURN.

ON X GOSUB Ordre de branchements multiples avec retour par RETURN. Marche comme ON X GOTO...

FOR I=0 TO 10 STEP 2 Répéter pour I allant de 0 à 10 et de 2 en 2 les instructions jusqu'au prochain **NEXT**.

REPEAT Répéter autant de fois les instructions qui suivent jusqu'au prochain **UNTIL** Jusqu'à ce que la condition qui suit **UNTIL** soit fausse.

STOP Interruption du programme.

END Arrêt du programme.

CALL X Ordre de branchement à un sous-programme codé en langage machine. Le retour se fait par RTS.

DEF FNA(x)=X+4 Définit une fonction numérique. A l'usage, X viendra se substituer dans l'expression écrite FNA(x).

POP Annule dernière adresse retour sur RETURN.

PULL Décale l'adresse d'un cran dans les boucles REPEAT (analogie avec POP).

Les constantes :

DATA X, "AA", Définit des éléments de données qui peuvent être utilisés par un READ.

READ A/A$.. Lit une donnée contenue dans la DATA.

RESTORE Repositionne la prochaine lecture par READ au début des DATAS.

PI Constante : = 3.14159265.

REM abcdef. Permet d'écrire des commentaires dans un programme. Ignorée à l'exécution.

Accès mémoire RAM

X=PEEK(Y) Lit l'octet d'adresse absolue Y et le range dans X.

X=DEEK(Y) Lit les 2 octets d'adresse absolue Y et les range dans X.

POKE Y, N Ecrit dans l'octet d'adresse absolue Y, la valeur N.

DOKE Y,N Ecrit dans les 2 octets d'adresse absolue Y la valeur de N.

USR (N) Transmet une valeur à un programme écrit en langage machine.

FRE(0) Provoque le nettoyage mémoire.

GRAB Utilisation mémoire de #9800 à #B499.

RELEASE Alloue à HIRES la mémoire de #9800 à #B499.

Fonctions mathématiques :

INT(X) Donne la partie entière de X.

RND(1) Donne un nombre aléatoire compris entre : 0 et 0.999999999999 à chaque utilisation.

SGN(X) Donne −1 si X Plus petit que 0, 0 si X=0 et +1 Si X est positif.

ABS(X) Donne la valeur absolue de X.

SQR(X) Donne la racine carrée de X.

EXP(X) Donne 2.718289 élevé à la puissance X.

LOG(X) Donne le Logarithme naturel de X.

LN(X) Donne le Logarithme Népérien (e).

SIN(X) Donne le SINUS de l'angle X en radians.

COS(X) Donne le COSINUS de l'angle X en radians.

TAN(X) Donne la tangente de l'angle X en radians.

ATN(X) Donne l'arc tangente en radian de X.

Sauvegarde/Lecture des programmes sur cassettes :

CSAVE"XX" Ecrit le programme présent en mémoire sur cassette. (Vitesse transfert=2400 bauds).

CSAVE"XX",S Idem mais vitesse transfert=300 bauds

CSAVE"XX",AUTO Sauvegarde avec RUN sur CLOAD.

CSAVE"XX",A#1000,E#1200 Sauvegarde sur cassette d'un bloc mémoire début(A)#1000, fin(E)#1200.

CLOAD"XX" Lecture d'un programme écrit sur cassette et chargement en mémoire RAM Basic. (2400).

CLOAD"XX",S Idem, mais Vitesse =300 bauds

CLOAD"XX",A#9800,E#9900 Lecture bloc, puis chargement en RAM debut(A)#9800, fin(E)#9900.

La couleur :

PAPER X Donne la couleur du fond.

INK X Donne la couleur des caractères.

0 = Noir	3 = Jaune	6 = Cyan
1 = Rouge	4 = Bleu	7 = Blanc
2 = Vert	5 = Magenta	

Contrôle du curseur :

TAB(N) Positionne le curseur de N espaces à partir du bord gauche de l'écran.

CLS Efface l'écran et renvoie le curseur en haut et à gauche.

SPC(N) Dans une instruction PRINT provoque N espace à l'écran.

X=SCRN(X,Y) Fournit le code ASCII du caractère affiché à l'écran à la position X(ligne) et Y(colonne).

TEXT Efface l'écran, et retourne au mode texte.

PLOT X,Y,"A" Ecrit un caractère page texte en X(ligne) Y(colonne).

PLOT X,Y,"ABCDE" Ecrit une chaîne de caractères en X et Y page Texte.

LORES Passe en mode basse définition(40 × 25)
LORES 0 Utilise le clavier standard.
LORES 1 Utilise le deuxième clavier.
Le fond devient Noir.

Les instructions du graphique :

HIRES Affiche la page haute résolution graphique (#A000 à #BFEO).

CURMOV X,Y,FD Déplacement relatif curseur
Le point est marqué en fonction de FD.

CURSET X, Y, FD Déplacement absolu curseur.
X compris entre 0 et 239,
Y compris entre 0 et 199

DRA X,Y,FD Trace un trait de a, b en a + x, b + y suivant FD

FILL B,A,N Remplit un certain nombre de zones A sur B rangées selon la commande N.

POINT(X,Y) Renvoie la couleur du PIXEL.
0 si pixel = couleur fond, 1/−1 devant.

FD 0 = Fond, 1 = Devant, 2 = inverse, 3 = Rien.

CIRCLE R,FD Trace un cercle de rayon R et suivant FD.

CHAR X,S,FD Ecrit caractère code ASCII = X, S = 0 clavier standard, S = 1 clavier 2 suivant FD.

Le Son et la Musique :

SOUND Canal, Période, Volume
Canal de 1 à 6
Période de 0 à n
Grand = Grâve
Volume de 0 à 15

MUSIC Canal, Octave, Note, Volume
Canal 1,2 ou 3
Octave de 0 à 7
Note 1 à 12 (1 = DO..12 = SI)
1/2 ton.
Volume de 0 à 15

PLAY Son, Bruit, Enveloppe, Durée
Son de 1 à 6
Bruit de 1 à 6
Enveloppe de 1 à 7
Durée de 0 à 32767

ZAPP, PING, SHOOT, EXPLODE
Sons préparés.

WAIT 50 Attente de 500 Millisecondes

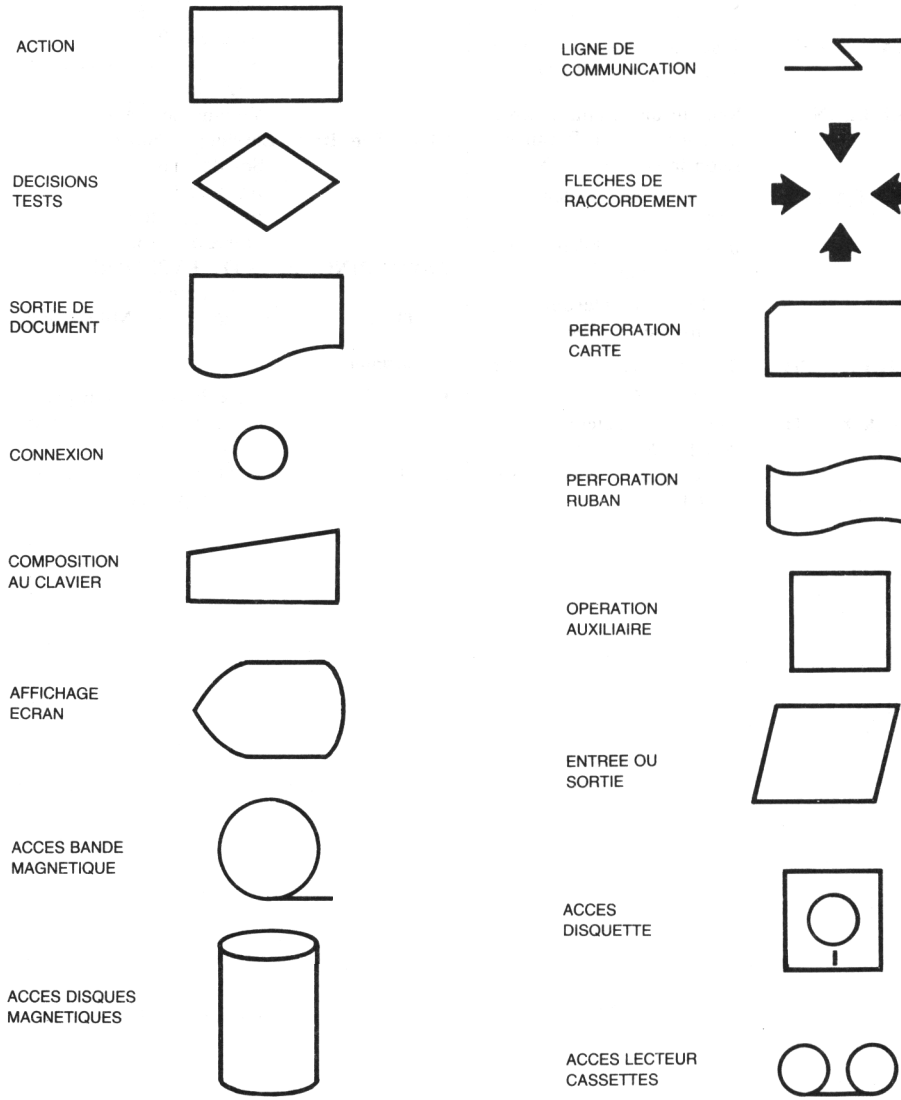
L'imprimante :

LLIST Liste du programme présent en mémoire sur l'imprimante

LPRINT.. Ecrit nombres et chaînes sur l'imprimante.

ANNEXE G DIFFÉRENTS SYMBOLES UTILISABLES POUR L'ORGANIGRAMME

Ces symboles se trouvent sur des ORGANIGRAPHES ou ORDINOGRAPHES (vendus en papeterie ou dans les boutiques d'ordinateurs).



Conclusion

J'espère, tout au long de ce livre, vous avoir convaincu qu'il n'est pas difficile d'utiliser et de programmer un ordinateur. S'il vous reste quelque appréhension, laissez-moi vous raconter comment j'ai fait la connaissance de l'ordinateur, à l'âge de douze ans.

Nous étions cernés par d'imposantes armoires métalliques. Le scintillement de quelques lampes témoins semblait leur donner vie. Là, un homme en blanc, les yeux rivés sur un écran vert, suivait attentif le déplacement sinusoïdal d'un spot lumineux. L'appareil était relié à une importante masse de fils électriques multicolores sortant d'un faux plancher. Plus loin, un autre homme regardait par transparence le filament d'une étonnante ampoule électrique. La dure lumière bleutée des néons emplissait cet univers de laboratoire d'avant-garde, très mystérieux pour moi. Nous traversâmes ainsi des salles presque toutes identiques séparées par des sas devant lesquels il fallait s'affranchir d'un contrôle. L'air était froid. J'eus le désagréable sentiment d'être un étranger en ces lieux. Cela contribua à faire croître mon étonnement naïf qui s'amplifia au cours de la visite.

Enfin nous pénétrâmes dans une salle plus spacieuse et mieux éclairée que les autres. Il y régnait une activité plus intense. Le crépitement d'une étonnante machine à écrire attira immédiatement mon attention. Je suivis, émerveillé, le déplacement d'un bras articulé, inscrivant une série de chiffres et de lettres sur un rouleau de papier. L'ensemble était parfaitement cadencé par la pénétration régulière d'une longue bande de papier aux perforations multiples. Machinalement, je lus des suites de mots que je ne pus comprendre. Il ne fallut pas plus de cinq minutes pour qu'une trentaine de lignes soient inscrites. Un homme, aux gestes pressés, vint découper la feuille, puis se dirigea rapidement vers un groupe de confrères impatients.

Après un court instant de réflexion générale, les visages affichèrent des grimaces de mécontentement. Certains allèrent consulter d'épais classeurs comme on va chercher la signification d'un mot étrange dans un dictionnaire. D'autres se grattaient tout simplement la tête et attendaient visiblement des explications salvatrices. D'autres encore discouraient dans un langage qui m'était tout à fait inconnu. Des mots comme « MEMOIRE, ACCU, PROGRAMME » s'élevaient des conversations. L'air très entendu de l'auditoire me plongea dans le plus profond respect que l'on a pour les grands hommes et me fit comprendre à quel point je pouvais être ignorant. Je suivais, passif, les mouvements de la grande salle. Je flânais dans mes rêveries où j'occupais la place du maître des lieux, intervenant fort à propos dans les conversations, expliquant l'inexplicable, comprenant l'incompréhensible. Le sifflement aigu d'un curieux appareil me fit sortir de mes songes. Au travers d'une vitre frontale se déroulait par à-coups un large ruban magnétique. L'ensemble me fit penser à un gros magnétophone. Des appareils semblables se mirent simultanément en marche. Je fus étonné par ce spectacle inhabituel qui s'arrêta brutalement.

Je fus plongé dans l'obscurité. Seuls, les plafonniers de secours laissaient filtrer une faible lumière jaunâtre. C'était l'affolement. Des gens couraient à droite, à gauche, actionnant des manettes, tirant des leviers. Plusieurs sonneries de téléphone tintèrent d'impatience. Des portes s'ouvrirent précipitamment laissant le passage à des hommes aux visages inquiets. Les télétypes se turent, les bobines des dérouleurs de bandes restèrent figées dans leurs derniers mouvements. L'oscilloscope, désespérément muet, refusa son diagnostic. Le silence s'imposa petit à petit. La respiration de la climatisation laissait deviner quelques restes de vie. Comment tant de puissance, de génie accumulé, pouvait-il se trouver réduit à l'état d'un inerte tas de ferraille ? L'instant était étrange, fascinant, presque inquiétant.

J'entendis bientôt une exclamation de soulagement général : l'appareil s'était remis en marche. L'inquiétude faisait place au sourire sur le visage des ingénieurs, et l'activité retrouva son rythme de croisière.

Ce calculateur était l'un des premiers en France à être opérationnel. Il s'appelait KL901 et appartenait à l'OFFICE NATIONAL METEOROLOGIQUE (O.N.M.). Nous étions au début des années 60 et, du haut de mes douze ans, je pensais avoir touché les sommets du progrès technologique. Plus tard, je compris qu'il n'en était rien et qu'en réalité j'étais le témoin des balbutiements d'une nouvelle science nommée aujourd'hui l'INFORMATIQUE. C'était le premier âge de l'ordinateur qui s'était mythiquement imposé à moi. Cette image de l'intouchable me poursuivit longtemps. Je sais aujourd'hui que l'informatique n'a rien de mystérieux ni d'inaccessible, et que l'apparition des ordinateurs personnels a beaucoup fait pour populariser auprès du grand public l'idée que tout le monde peut s'en approcher, s'il le veut bien.

Georges VIGUIER.

LA BIBLIOTHÈQUE EDIMICRO

- *Collection « Guides microordinateurs »*

de Merly	**GUIDE DE L'APPLE**
	L'APPLE standard
	Les extensions
	Les applications
	Les langages
Bayvejiel	**GUIDE DE L'ORIC**
Bieber, Perbost, Renucci	**GUIDE DU TO 7**

- *Collection « Jeux »*

Chane-Hune, Darbois	**JEUX SUR ORIC**
Perbost, Renucci	**JEUX SUR TO 7**

- *Collection « Logiciels »*

Bonnet, Dinh	**MULTIPLAN SUR APPLE**
	Exercices de Gestion

- *Ouvrages de base*

Viguier	**Premiers pas en programmation sur ORIC**

Les séminaires de formation FDS/Edimicro

- *Pour utilisateurs non spécialistes*

  VISICALC - MULTIPLAN - GESTION DE FICHIERS - BASES DE DONNÉES - CHOIX D'UN MICROORDINATEUR - BUREAUTIQUE...

- *Pour programmeurs débutants*

  PREMIERS PAS EN BASIC - FICHIERS EN BASIC...

- *Pour professionnels de l'informatique*

  TÉLÉMATIQUE - UNIX - XENIX - MS-DOS - LANGAGE C...

EDIMICRO
121-127, avenue d'Italie, 75013 PARIS

EDIMICRO
121-127, avenue d'Italie, 75013 PARIS

Imprimé en France. — JOUVE, 18, rue Saint-Denis, 75001 PARIS
N° 11949. Dépôt légal : Décembre 1983

PREMIERS PAS EN PROGRAMMATION SUR ORIC

Georges Viguier

"Je connais Georges Viguier depuis longtemps. Je connais donc son talent pour faire partager ses connaissances au plus grand nombre.
Ce livre en est la preuve : les exemples sont clairs et bien choisis, la méthode simple et rationnelle.
Vous pourrez réaliser rapidement vos premiers programmes."

L. Augustoni
Rédacteur en Chef de MICR'ORIC
Revue des utilisateurs de l'ORIC

Tout le monde peut s'initier à la programmation des microordinateurs.
Ce livre vous donne trois atouts pour bien comprendre :

- Le langage retenu est le **BASIC,** considéré comme le plus adapté aux débutants et qui présente l'avantage d'être disponible sur tous les microordinateurs.
- La méthode d'approche est originale, **accessible à tous :** les exemples sont tirés de la vie courante, les démonstrations sont appuyées par de nombreux schémas et figures.
- Les programmes sont réalisés sur un **ORIC,** qui convient parfaitement à cet apprentissage, et qui est très représentatif des microordinateurs de sa catégorie. Les connaissances acquises vous serviront sur toute autre machine du même type.

Vous apprendrez donc, *en douceur,* à réaliser de petits programmes – programmes de jeux notamment – et vous aurez les bases pour aller plus loin...

Edimicro

121-127 Avenue d'Italie 75013 Paris

ISBN 2-904457-07-0

PRIX 79 F

Edimicro **PREMIERS PAS EN PROGRAMMATION SUR ORIC** Georges VIGUIER